

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In re Patent Application of

Gyeong-ja JANG et al.

Group Art Unit: 2176

Application No.: 10/727,659

Examiner: Unassigned

Filing Date: December 5, 2003

Confirmation No.: 8978

Title: METHOD AND SYSTEM FOR GENERATING INPUT FILE USING META LANGUAGE REGARDING  
GRAPHIC DATA COMPRESSION

**SUBMISSION OF CERTIFIED COPY OF PRIORITY DOCUMENT**

Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Sir:

The benefit of the filing date of the following priority foreign application(s) in the following foreign country is hereby requested, and the right of priority provided in 35 U.S.C. § 119 is hereby claimed.

Country: Republic of Korea

Patent Application No(s): 10-2003-0084216

Filed: November 25, 2003

In support of this claim, enclosed is a certified copy(ies) of said foreign application(s). Said prior foreign application(s) is referred to in the oath or declaration. Acknowledgment of receipt of the certified copy(ies) is requested.


Respectfully submitted,

BURNS, DOANE, SWECKER & MATHIS, L.L.P.

P.O. Box 1404  
Alexandria, Virginia 22313-1404  
(703) 836-6620

Date: April 20, 2004

By

  
\_\_\_\_\_  
Charles F. Wieland III  
Registration No. 33,096



별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출원 번호 : 10-2003-0084216  
Application Number

출원 년 월 일 : 2003년 11월 25일  
Date of Application NOV 25, 2003

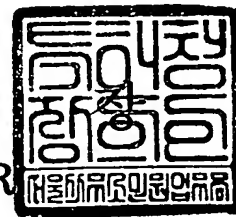
출원인 : 삼성전자주식회사  
Applicant(s) SAMSUNG ELECTRONICS CO., LTD. 20638



2003 년 12 월 02 일

특 허 청

COMMISSIONER



【서지사항】

【서류명】	특허출원서
【권리구분】	특허
【수신처】	특허청장
【참조번호】	0036
【제출일자】	2003.11.25
【국제특허분류】	G06F
【발명의 명칭】	그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성 방법 및 시스템
【발명의 영문명칭】	Method and system for generation input file using meta language regarding graphic data compression
【출원인】	
【명칭】	삼성전자 주식회사
【출원인코드】	1-1998-104271-3
【대리인】	
【성명】	이영필
【대리인코드】	9-1998-000334-6
【포괄위임등록번호】	2003-003435-0
【대리인】	
【성명】	이해영
【대리인코드】	9-1999-000227-4
【포괄위임등록번호】	2003-003436-7
【발명자】	
【성명의 국문표기】	장경자
【성명의 영문표기】	JANG,Gyeong Ja
【주민등록번호】	750405-2232819
【우편번호】	412-719
【주소】	경기도 고양시 덕양구 행신동 햇빛마을 1901동 1201호
【국적】	KR
【발명자】	
【성명의 국문표기】	김도균
【성명의 영문표기】	KIM,Do Kyoon
【주민등록번호】	690605-1041815

【우편번호】	463-030
【주소】	경기도 성남시 분당구 분당동 131-3번지 3층
【국적】	KR
【우선권주장】	
【출원국명】	US
【출원종류】	특허
【출원번호】	60/430,977
【출원일자】	2002.12.05
【증명서류】	미첨부
【우선권주장】	
【출원국명】	US
【출원종류】	특허
【출원번호】	00/000,000
【출원일자】	2003.10.14
【증명서류】	미첨부
【심사청구】	청구
【취지】	특허법 제42조의 규정에 의한 출원, 특허법 제60조의 규정에 의한 출원심사를 청구합니다. 대리인 이영필 (인) 대리인 이해영 (인)
【수수료】	
【기본출원료】	20 면 29,000 원
【가산출원료】	63 면 63,000 원
【우선권주장료】	2 건 43,000 원
【심사청구료】	21 항 781,000 원
【합계】	916,000 원
【첨부서류】	1. 요약서·명세서(도면)_1통 2.우선권증명서류 및 동 번역문_2통

## 【요약서】

### 【요약】

본 발명은 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성 방법 및 시스템에 관한 것으로서, 그 방법은 적어도 압축할 객체 데이터에 관한 정보를 포함하는 압축노드와, 압축에 필요한 매개인자를 정의하고 있는 XML 스키마를 구비하는 단계; XML 입력파일을 상기 XML 스키마를 참조하여 소정의 데이터 압축부호화기에 입력되는 파일로의 변환을 지원하는 스타일 쉬트(style sheet)를 구비하는 단계; 및 XML 입력파일을 상기 XMT 스키마 및 상기 스타일 쉬트를 참조하여 파싱하여 소정의 데이터 압축부호화기에 입력되는 파일을 생성하는 단계를 포함함을 특징으로 한다. 본 발명에 의하면, 저작자가 메타 표현 방법을 사용해서 3D 콘텐츠 저작단계에서 3D 데이터의 표현 및 압축할 수 있다. 저작자가 압축의 선택과 조절 가능하다. 그 방법은 메타 표현의 선택 여부에 따라 압축의 가능 여부를 결정할 수 있다. 또한 메타 표현 방법을 조절하므로 압축 방식을 조절할 수 있다

### 【대표도】

도 2

【명세서】

【발명의 명칭】

그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성 방법 및 시스템{Method and system for generation input file using meta language regarding graphic data compression}

【도면의 간단한 설명】

도 1은 본 발명에 의한 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성시스템의 구성을 블록도로 도시한 것이다.

도 2는 본 발명에 의한 그래픽 압축에 관한 메타표현을 이용한 입력파일 생성시스템의 다른 구성을 블록도로 도시한 것이다.

도 3은 텍스트 구문(textual syntax)을 사용해서 MPEG-4 장면 기술(scene description)을 나타내기 위한 XMT 프레임워크이다. 도 4는 물체 A에 대한 3차원 데이터를 매개인자를 사용해 압축하여 물체 A의 압축된 비트스트림을 버퍼("BufferWithEP.m3d")를 사용해서 전송하는 방법을 나타낸다.

도 5는 이미 압축된 비트스트림을 버퍼로 전송하는 경우를 설명하는 도면이다.

도 6은 BitWrapper 노드 안에서 물체 A에 대한 3차원 데이터를 매개인자를 사용해 압축하여 물체 A의 압축된 비트스트림을 URL을 사용해 전송하는 방법을 나타낸다.

도 7은 BitWrapper 노드 안에서 이미 압축된 물체 A에 대한 3차원 데이터의 비트스트림을 URL을 사용해 전송하는 방법을 나타낸다.

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

- <7> 본 발명은 그래픽 데이터 저작에 관한 것으로서, 특히 그래픽 데이터 압축에 관한 메타 표현을 이용한 입력파일 생성 방법 및 시스템에 관한 것이다.
- <8> 종래의 확장가능한 MPEG-4 텍스트 포맷(Extensible MPEG-4 Textual Format :이하 XMT라 한다.) 기술은 MPEG-4 프리미티브(2D, 3D 그래픽, 오디오, 비디오 등)에 대해 저작자 입장에서 쉽고 편리하게 다룰 수 있는 표현 방법에 대한 것이었다. 또한 저작자가 만든 데이터가 여러 애플리케이션에서 재사용될 수 있고, 데이터의 호환성과 이식성이 있도록 콘텐츠 저작에 대한 프레임워크(framework)를 만들었다. 이것은 기존 XMT 기술이 MPEG-4 프리미티브(primitive)에 대한 XML 구문(syntax)을 정의하였기에 가능했다. 그러나 종래의 XMT는 3D 데이터의 압축표현에 대해서는 다루고 있지 않다. 그래서 저작자가 만든 3D 콘텐츠에 대한 애니메이션 데이터 및 표현 데이터에 대한 압축이 필요할 경우 압축할 수가 없었다.

【발명이 이루고자 하는 기술적 과제】

- <9> 본 발명이 이루고자 하는 기술적 과제는, 저작단계에서 그래픽 데이터의 압축을 쉽게 다룰 수 있도록 하기 위해, 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성 방법 및 시스템을 제공하는 것이다. 또한 저작단계에서 그래픽 데이터의 압축을 쉽게 다룰 수 있도록 하기 위해, MPEG-4 AFX에서 제안한 압축 표현을 XMT로 정의하여 그래픽 데이터를 압축할 수 있게 하는, 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성 방법 및 시스템을 제공하는 것이다.

# 【발명의 구성 및 작용】

<10>       상기 기술적 과제를 이루기 위한 본 발명에 의한 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성 방법은, 적어도 압축할 객체 데이터에 관한 정보를 포함하는 압축노드와, 압축에 필요한 매개인자를 정의하고 있는 XML 스키마를 구비하는 단계; XML 입력파일을 상기 XML 스키마를 참조하여 소정의 데이터 압축부호화기에 입력되는 파일로의 변환을 지원하는 스타일 쉬트(style sheet)를 구비하는 단계; 및 XML 입력파일을 상기 XML 스키마 및 상기 스타일 쉬트를 참조하여 파싱하여 소정의 데이터 압축부호화기에 입력되는 파일을 생성하는 단계를 포함함을 특징으로 한다. 상기 XML 스키마는 적어도 상기 압축된 객체 데이터를 저장하고 있는 파일의 위치정보를 포함하고 있는 EncodingHints를 더 구비함이 바람직하다. 상기 압축매개인자는 압축 대상이 되는 객체의 정점 좌표에 대한 키프레임 애니메이션 데이터에 관한 매개인자, 3차원 메쉬 정보에 관한 매개인자, 회전 이동 키프레임 애니메이션 데이터에 관한 매개인자 및 위치 이동 키프레임 애니메이션 데이터에 관한 매개인자 중 적어도 하나를 포함함이 바람직하다.

<11>       상기 기술적 과제를 이루기 위한 본 발명에 의한, MPEG-4 AFX에서 제안한 압축 표현을 XMT로 정의하여 그래픽 데이터를 압축할 수 있게 하는 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성 방법은, 압축할 객체 데이터에 관한 정보를 포함하고 있는 압축노드와, 압축에 필요한 매개인자, 및 적어도 압축된 객체 데이터 파일의 위치정보를 포함하는 BitWrapperEncodingHints를 정의하고 있는 XMT 스키마를 구비하는 스키마단계; XMT 입력파일을 상기 XMT 스키마를 참조하여 scene 파일로의 변환을 지원하는 XMT2BIFS style sheet와, XMT 입력파일을 상기 XMT 스키마를 참조하여 mux 파일로의 변환을 지원하는 XMT2MUX style sheet를 구비하는 스타일쉬트단계; 및 XMT 입력화일을 상기 XMT 스키마와 상기 XMT2BIFS style sheet

및 XMT2MUX style sheet를 이용하여 파싱하여 scene 파일 및 mux 파일을 생성하는 파싱단계를 포함함을 특징으로 한다. 상기 압축노드는 압축할 객체 데이터를 포함하고 있는 노드필드; url 필드와 상호 배타적으로 사용되며, 상기 노드의 압축된 비트스트림을 인-밴드로서 전송하는 buffer필드; 및 상기 buffer 필드와 상호 배타적으로 사용되며, 상기 노드의 압축된 비트스트림을 아웃-밴드로서 전송하는 url 필드를 포함하여 이루어짐이 바람직하다. 상기 압축노드는 노드압축 스킴의 종류를 나타내는 타입필드를 더 구비함이 바람직하다. 상기 압축매개인자는 압축 대상이 되는 객체의 정점 좌표에 대한 키프레임 애니메이션 데이터에 관한 매개인자, 3차원 메쉬 정보에 관한 매개인자, 회전 이동 키프레임 애니메이션 데이터에 관한 매개인자 및 위치 이동 키프레임 애니메이션 데이터에 관한 매개인자 중 적어도 하나를 포함함이 바람직하다. 상기 BitWrapperEncodingHints는 압축노드의 URL ID 와 같은 객체기술 ID정보 및 mux 파일이 제공하는 압축된 비트스트림을 전송하는 파일이름과 비트스트림 포맷 종류에 관한 정보를 더 구비함이 바람직하다. 상기 파싱단계는 원본데이터와 압축 매개인자 및 버퍼정보를 구비하는 압축노드를 포함하는 XMT 입력파일을 받아들이는 단계; 및 상기 XMT 스키마와 상기

XMT2BIFS

style sheet 및 XMT2MUX style sheet를 이용하여 상기 XMT 입력파일을 파싱하여 scene 파일 및 mux 파일을 생성하는 단계를 포함하여 이루어짐이 바람직하고, 상기 scene 파일은 상기 원본데이터와, 압축 매개인자와, 상기 원본데이터를 압축한 비트스트림을 전송할 버퍼 정보를 구비하며, 상기 mux 파일은 상기 scene 파일을 BIFS 인코더를 통해 부호화된 파일이름과, 스트림 포맷 정보를 구비한다. 상기 파싱단계는 이미 압축된 객체데이터가 저장되어 있는 버퍼정보를 구비하는 압축노드를 포함하는 XMT 입력파일을 받아들이는 단계; 및 상기 XMT 스키마와 상기 XMT2BIFS style sheet 및 XMT2MUX style sheet를 이용하여 상기 XMT 입력파일을 파싱하여 scene 파일 및 mux 파일을 생성하는 단계를 포함하여 이루어짐이 바람직하다. 상기 scene 파일은 객체데이터를 압축한 비트스트림을 전송할 버퍼 정보를 구비하며, 상기 mux 파일은 상기 scene 파일을 BIFS 인코더를 통해 부호화된 파일이름과, 스트림 포맷 정보를 구비한다. 상기 파싱단계는 원본데이터와 압축 매개인자 및 url정보를 구비하는 압축노드와, 압축노드의 url ID 와 같은 객체기술 ID(objectDescriptor ID) 정보 및 객체의 압축된 비트스트림의 위치정보를 구비하는 BitWrapperEncodingHints를 포함하는 XMT 입력파일을 받아들이는 단계; 및 상기 XMT 스키마와 상기 XMT2BIFS style sheet 및 XMT2MUX style sheet를 이용하여 상기 XMT 입력파일을 파싱하여 scene 파일 및 mux 파일을 생성하는 단계를 포함하여 이루어짐이 바람직하고, 상기 scene 파일은 상기 원본데이터와, 압축 매개인자와, 상기 원본데이터를 압축한 비트스트림을 전송할 url 정보를 구비하며, 상기 mux 파일은 상기 BitWrapperEncodingHints에 명시된 객체의 압축된 비트스트림의 위치정보

및 스트림 포맷 정보를 구비한다. 상기 XMT 입력파일은 상기 BitWrapperEncodingHints에 나타난 객체 ID와 동일한 객체 기술 ID 정보와 파싱 결과 생성될 mux 파일이름 정보를 포함하고 있는 ObjectDescriptorUpdate를 더 구비하며, 상기 scene 파일은 상기 BitWrapperEncodingHints에 나타난 객체 ID와 동일한 객체기술 ID 정보와 mux 파일 이름 정보를 더 구비함이 바람직하다.

<12>       상기 파싱단계는 이미 압축된 객체데이터가 링크된 URL 정보를 구비하는 압축노드와 URL ID와 같은 객체기술(objectDescriptor) ID 정보 및 객체의 압축된 비트스트림의 위치정보를 구비하는 BitWrapperEncodingHints 를 포함하는 XMT 입력파일을 받아들이는 단계; 및 상기 XMT 스키마와 상기 XMT2BIFS style sheet 및 XMT2MUX style sheet를 이용하여 상기 XMT 입력파일을 파싱하여 scene 파일 및 mux 파일을 생성하는 단계를 포함하여 이루어짐이 바람직하고, 상기 scene 파일은 상기 압축노드에 명시된 객체기술 ID와 동일하며 상기 원본데이터를 압축한 비트스트림을 전송할 url 정보를 구비하며, 상기 mux 파일은 상기 BitWrapperEncodingHints에 명시된 객체의 압축된 비트스트림의 위치정보 및 스트림 포맷 정보를 구비한다. 상기 XMT 입력파일은 상기 BitWrapperEncodingHints에 나타난 객체 ID와 동일한 객체 기술 ID 정보와 파싱 결과 생성될 mux 파일이름 정보를 포함하고 있는 ObjectDescriptorUpdate를 더 구비하며, 상기 scene 파일은 상기 BitWrapperEncodingHints에 나타난 객체 ID와 동일한 객체 기술 ID 정보와 mux 파일이름 정보를 더 구비함이 바람직하다. 상기 기술적 과제를 이루기 위한 본 발명에 의한 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성 시스템은, 적어도 압축할 객체 데이터에 관한 정보를 포함하는 압축노드와, 압축에 필요한 매개인자를 정의하고 있는 XML 스키마;

<13>       XML 입력파일을 상기 XML 스키마를 참조하여 소정의 데이터 압축부호화기에 입력되는 파일로의 변환을 지원하는 스타일 쉬트(style sheet); 및 XML 입력파일을 상기 XMT 스키마 및 상

기 스타일 쉬트를 참조하여 파싱하여 소정의 데이터 압축부호화기에 입력되는 파일을 생성하는 XLM파서를 포함함을 특징으로 한다. 상기 압축매개인자는 압축 대상이 되는 객체의 정점 좌표에 대한 키프레임 애니메이션 데이터에 관한 매개인자, 3차원 메쉬 정보에 관한 매개인자, 회전 이동 키프레임 애니메이션 데이터에 관한 매개인자 및 위치 이동 키프레임 애니메이션 데이터에 관한 매개인자 중 적어도 하나를 포함함이 바람직하다. 상기 기술적 과제를 이루기 위한 본 발명에 의한, MPEG-4 AFX에서 제안한 압축 표현을 XMT로 정의하여 그래픽 데이터를 압축할 수 있게 하는 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성 시스템은, 압축할 객체 데이터에 관한 정보를 포함하고 있는 압축노드와, 압축에 필요한 매개인자, 및 적어도 압축된 객체 데이터 파일의 위치정보를 포함하는 BitWrapperEncodingHints를 정의하고 있는 XMT 스키마; XMT 입력파일을 상기 XMT 스키마를 참조하여 scene 파일로의 변환을 지원하는 XMT2BIFS 스타일 쉬트; XMT 입력파일을 상기 XMT 스키마를 참조하여 mux 파일로의 변환을 지원하는 XMT2MUX 스타일 쉬트; 및 XMT 입력파일을 상기 XMT 스키마와 상기 XMT2BIFS style sheet 및 XMT2MUX 스타일 쉬트를 이용하여 파싱하여 scene 파일 및 mux 파일을 생성하는 XMT파서를 포함함을 특징으로 한다. 상기 압축노드는 압축할 객체 데이터를 포함하고 있는 노드필드; url 필드와 상호 배타적으로 사용되며, 상기 노드의 압축된 비트스트림을 인-밴드로서 전송하는 buffer필드; 및 상기 buffer 필드와 상호 배타적으로 사용되며, 상기 노드의 압축된 비트스트림을 아웃-밴드로서 전송하는 url 필드를 포함하여 이루어짐이 바람직하다. 상기 압축매개인자는 압축 대상이 되는 객체의 정점 좌표에 대한 키프레임 애니메이션 데이터에 관한 매개인자, 3차원 메쉬 정보에 관한 매개인자, 회전 이동 키프레임 애니메이션 데이터에 관한 매개인자 및 위치 이동 키프레임 애니메이션 데이터에 관한 매개인자 중 적어도 하나를 포함함이 바람직하다. 상기 BitWrapperEncodingHints는 압축노드의 URL ID와 같은 객체기술(objectDescriptor)

ID 정보 및 mux 파일이 제공하는 압축된 비트스트림을 전송하는 파일이름과 비트스트림 포맷 종류에 관한 정보를 더 구비함이 바람직하다.

- <14> 그리고 상기 기재된 발명을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체를 제공한다.
- <15> 이하, 첨부된 도면들을 참조하여 본 발명에 따른 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성시스템에 대해 상세히 설명한다.
- <16> 저작자가 3차원 데이터의 압축을 쉽게 다룰 수 있게 하기 위해서는 저작자가 3차원 콘텐츠를 저작하는 단계에서부터 3차원 데이터를 표현, 조작 및 압축하는데 필요한 여러 인자들을 쉽게 조절할 수 있는 방법이 필요하다. 이것은 XMT를 통해서 가능하다. 상기 XMT는 오디오, 비디오, 2차원 그래픽 및 3차원 그래픽과 같은 MPEG-4 콘텐츠 저작에 대한 프레임워크(framework)이다. 또한 상기 XMT는 텍스트 구문(textual syntax)을 사용해서 MPEG-4 장면 기술(scene description)을 나타내기 위한 프레임워크이다. 상기 프레임워크는 도 3에 잘 나타나 있다. XMT는 콘텐츠 저작자에게 저작자가 만든 콘텐츠(contents)를 다른 저작자, 다른 툴(tool) 및 다른 서비스 공급자(provider)가 재사용 할 수 있게 해 준다. 그리고 X3D(Extensible 3D)와 SMIL(Synchronized Multimedia Integration Language)과도 상호 운용할 수 있다.
- <17> 도 3에서와 같이 XMT 포맷은 SMIL 플레이어, VRML 플레이어, MPEG-4 플레이어에서 상호 교환이 가능하고 플레이된다. 이를 보다 자세히 설명하면 XMT 포맷은 파싱(parsing)되어 SMIL 플레이어에서 플레이된다. 또한 XMT 포맷은 X3D에 대해 전처리를 수행한 다음 VRML 플레이어에서 플레이된다. 또한 MPEG-4 표현(mp4)에 대해 컴파일한 후 MPEG-4 플레이어에서 플레이된다.

<18> XMT는 2단계 구조 즉, XMT-A 포맷과 XMT- OMEGA 포맷으로 구성된다. 상기 XMT-A는 MPEG-4 콘텐츠(오디오, 비디오, 2D, 3D 그래픽 데이터 표현 및 압축 등)의 XML 기반 버전(version)이고 확장된 3D 그래픽(X3D)을 포함한다. 또한 XMT-A에 포함된 것은 MPEG-4 특징을 나타내기 위해 X3D로 MPEG-4를 확장한 것이다. XMT-A는 텍스트포맷(textual format)과 이진포맷(binary format)을 1:1 매핑한다. XMT- OMEGA 는 MPEG-4 특징에 대한 상위 레벨 표현이고 SMIL을 기반으로 한다. XMT는 콘텐츠 저작자가 OMEGA 에서 A로의 메커니즘에 대해 몰라도 OMEGA 에서 A로 디폴트 매핑 가능하다. XMT- OMEGA 는 사용자에게 저작하기 쉽고 편리한 인터페이스 표현 기능을 제공한다. 실제 MPEG-4 데이터에 대한 표현, 처리 및 압축에 대한 것은 XMT-A에서 이루어진다.

<19> 따라서 저작자가 3D 데이터를 압축하려면 저작자가 3D 콘텐츠를 저작하는 단계에서부터 3D 데이터를 표현, 조작 및 압축하는데 필요한 여러 인자들을 쉽게 조절하도록 압축에 관련된 기술을 XMT-A로 정의하고 이를 사용해서 데이터를 압축할 수 있게 하는 것이다.

<20> 즉 저작자가 만든 3D 콘텐츠에 대한 애니메이션 데이터 및 표현 데이터에 대한 압축 기능은 MPEG-4 AFX에서 제안한 압축 표현을 XMT로 정의하므로 가능하다. 따라서 저작자가 압축 표현을 통해 3D 데이터를 압축해서 데이터를 전송할 수 있게 하면 된다. 이는 3D(애니메이션 및 표현)데이터를 압축하는데 필요한 여러 인자들을 XMT를 사용해 매개인자(parameter)로 정의하므로써 가능하다. 따라서 본 발명에서는 저작자가 압축을 표현하는 노드를 사용해서 3D 데이터를 압축할 때 필요한 여러 인자들을 XMT-A 스키마로 정의한다. 본 발명에서는 3D 데이터를 압축하는데 사용되는 여러 인자들에 대한 메타표현 방법을 제공하고 이를 사용하여 압축을 실행한다.

<21> 도 1은 본 발명에 의한 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성시스템의 구성을 블록도로 도시한 것으로서, XML 스키마(120), 스타일 쉬트(style sheet, 130) 및 XML파서(110)를 포함하여 이루어진다. 상기 XML 스키마(120)는 적어도 압축할 객체 데이터에 관한 정보를 포함하는 압축노드와, 압축에 필요한 매개인자를 정의하고 있다. 상기 스타일 쉬트(style sheet, 130)는 XML 입력파일을 상기 XML 스키마(120)를 참조하여 데이터 압축부호화기(140)에 입력되는 파일로의 변환을 지원한다. 상기 XML파서(110)는 XML 입력파일(100)을 상기 XMT 스키마(120) 및 상기 스타일 쉬트(130)를 참조하여 파싱하여 데이터 압축부호화기(140)에 입력되는 파일을 생성한다.

<22> 도 2는 도 1에 도시된 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성시스템의 일 예에 해당하는 것으로서, MPEG-4 AFX에서 제안한 압축 표현을 XMT로 정의하여 그래픽 데이터를 압축할 수 있게 하는 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성시스템의 구성을 블록도로 도시한 것이다.

<23> 상기 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성 시스템은 XMT 스키마(240), XMT2BIFS 스타일 쉬트(230), XMT2MUX 스타일 쉬트(220), XMT파서(210)를 포함하여 이루어진다. 상기 XMT 스키마(240)는 압축할 객체 데이터에 관한 정보를 포함하고 있는 압축노드와, 압축에 필요한 매개인자, 및 적어도 압축된 객체 데이터 파일의 위치정보를 포함하는 BitWrapperEncodingHints를 정의하고 있다. XMT2BIFS 스타일 쉬트(230)는 XMT 입력파일(200)을 상기 XMT 스키마(210)를 참조하여 scene 파일로의 변환을 지원한다. XMT2MUX 스타일 쉬트(220)는 XMT 입력파일을 상기 XMT 스키마를 참조하여 mux 파일로의 변환을 지원한다. XMT파서(210)는 XMT 입력화일을 상기 XMT 스키마와 상기 XMT2BIFS style sheet 및 XMT2MUX style sheet를 이용하여 파싱하여 scene 파일 및 mux 파일을 생성한다. 상기 압축노드는 압축할 객체

데이터를 포함하고 있는 노드필드, url 필드와 상호 배타적으로 사용되며, 상기 노드의 압축된 비트스트림을 인-밴드로서 전송하는 buffer필드 및 상기 buffer 필드와 상호 배타적으로 사용되며, 상기 노드의 압축된 비트스트림을 아웃-밴드로서 전송하는 url 필드를 구비한다. 그리고 상기 압축매개인자는 압축 대상이 되는 객체의 모든 정점좌표에 대한 키프레임 애니메이션 데이터에 관한 매개인자, 3차원 메쉬 정보에 관한 매개인자, 회전이동 키프레임 애니메이션 데이터에 관한 매개인자 및 위치이동 키프레임 애니메이션 데이터에 관한 매개인자 중 적어도 하나를 선택할 수 있다. 상기 BitWrapperEncodingHints는 압축노드의 URL ID 와 같은 객체기술 ID(objectDescriptor ID) 정보 및 mux 파일이 제공하는 압축된 비트스트림을 전송하는 파일 이름과 비트스트림 포맷 종류에 관한 정보를 더 구비한다.

<24> 도 2에 도시된 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성 시스템을 보다 상세히 설명한다. 먼저 3D 데이터를 압축하는 노드와 그 매개 인자의 XMT-A 스키마를 사용하여 압축을 실행하는 방법 및 구조를 설명하기로 한다.

<25> 기존 XMT 기술은 3D 데이터를 압축하는 표현에 대한 XMT-A 스키마가 정의되지 않았으므로 3D 데이터 압축에 대한 XMT 파일이 입력으로 들어와도 파싱할 수가 없었다. 그러나 도 2에 도시된 바와 같이 3D 데이터를 압축하는 노드와 그 매개인자(parameter)를 XMT-A 스키마(240)에 정의해 주므로써 압축노드를 사용한 3D 데이터 압축에 대한 XMT 파일(200)이 입력으로 들어왔을 때 XMT 파서(parser, 210)가 XMT-A 스키마(240)에 정의된 내용과 XMT2MUX 스타일 쉬트(220)와 XMT2BIFS 스타일 쉬트(230)를 바탕으로 파싱하고, MPEG-4 표준 부호화기의 입력파일을 만들어 낸다. MPEG-4 표준 부호화기는 BIFS부호화기(250)와 MP4 부호화기(260)로 구성된다. 만들어진 입력파일이 각각 BIFS 부호화기(250)와 MP4 부호화기(260)로 들어가 비트스트림(.mp4)을 만들어 내고, MPEG-4 플레이어(Player)에서 시각화되어 사용자에게 보여진다.

- <26> 3D 데이터를 압축하는 노드와 그 매개인자에 대한 메타 표현, 즉 XMT-A 스키마를 사용하면 저작 단계에서 저작자가 메타 표현의 선택 여부에 따라 압축을 할 수도 있고 압축을 사용하지 않을 수도 있다. 또한 압축을 선택할 경우 압축하고자 하는 3D 데이터에 대해 압축 매개 인자에 메타표현을 할 수 있다.
- <27> 저작자가 압축을 선택할 경우 압축하고자 하는 3D 데이터를 그 매개 인자를 조절해서 크게 2가지 방법으로 선택해서 전송할 수 있다. 첫째는 원본 데이터를 압축하여 비트스트림으로 전송하는 방법과 둘째는 이미 압축된 비트스트림을 포함하여 전송하는 방법이 있다.
- <28> 상기 2가지 방법은 더 세분화되어 총 4가지의 전송 방법이 있으며, 저작자는 상기 4가지 전송방법을 선택할 수 있다. 첫번째 방법은 원본 데이터를 매개인자를 사용해 비트스트림으로 만들어 "buffer"를 사용하여 전송하는 방법이다. 두번째 방법은 이미 압축된 비트스트림 형태로 "buffer"를 사용해 전송하는 방법이다. 세번째 방법은 원본 데이터를 매개인자를 사용해 비트스트림으로 만들어 "url"로 전송하는 방법이다. 네번째 방법은 이미 압축된 비트스트림 형태로 "url"를 사용해 전송하는 방법이다.
- <29> 다음으로 3D 데이터 압축에 사용되는 여러 인자들에 대한 메타 표현 방법(XMT)을 설명하기로 한다. 상기 메타표현 방법은 3D 데이터를 압축하는데 필요한 노드와 그 매개인자를 메타 표현하는 방법이다. 여기서 3D 데이터를 압축하는데 필요한 노드인 BitWrapper 노드와 그 매개 변수를 실례로 들어 설명한다.
- <30> 1. BitWrapper 노드에 대한 XMT-A 스키마
- <31> 1.1 BitWrapper 노드의 BIFS 구문(Syntax)
- <32> BitWrapper { ##NDT=SF2DNode,SF3DNode,SFGeometryNode

<33> field SFWorldNode node NULL

<34> field SFInt32 type 0

<35> field MFUrl url []

<36> field SFString buffer ""}

<37> 우선 BitWrapper에 대해서 간략히 설명하면 다음과 같다. BitWrapper 노드의 기능은 "node" 필드의 데이터를 압축하고 이 비트스트림을 인-밴드(in-band) 또는 아웃-밴드(out-band)를 사용해서 전송하는 것이다. "url" 필드는 아웃-밴드 비트스트림을 전송하고, "buffer" 필드는 BIFS 비트스트림과 같이 인-밴드 비트스트림을 전송한다.

<38> 만약 저작자가 데이터를 압축해서 BitWrapper 노드를 통해 전송하고자 한다면 저작자는 비트스트림을 생성하기 위한 매개인자를 조절할 수 있어야 한다. 이것은 XMT-A 스키마 구문(Syntax)으로 할 수 있다. 그러나 이 매개인자는 특정 인코더에서 사용되어지는 파라미터가 아니다. 대신에 이 매개인자들은 디코더 구문(syntax)에 관련이 있다. 이 파라미터들은 압축 동안 조정될 수 있다. 현재 BitWrapper 노드는 7개의 압축 톨에 대한 압축된 비트스트림의 전송을 지원한다.

<39> 다음은 BitWrapper 노드와 3개의 3차원 키프레임 애니메이션 데이터 (CoordinateInterpolator, OrientationInterpolator, PositionInterpolator) 노드의 매개인자와 3차원 메쉬 정보(IndexedFaceSet) 노드의 매개인자에 대한 XMT-A 스키마 구문을 설명한다.

<40> 1.2 *BitWrapper* 노드에 대한 XMT-A 스키마

<41> 1.2.1 Syntax

<42> <?xml version="1.0" encoding="UTF-8"?><schema xmlns="http://www.w3.org/2001/XMLSchema"

```
<43>      xmlns:xmta="urn:mpeg:mpeg4:xmta:schema:2002"

<44>      targetNamespace="urn:mpeg:mpeg4:xmta:schema:2002"

<45>      elementFormDefault="qualified">

<46> <element name="BitWrapper">

<47>   <complexType>

<48>   <element name="node" minOccurs="0"   form="qualified">

<49>     <complexType>

<50>       <group ref="xmta:SFWorldNodesType" minOccurs="0"/>

<51>     </complexType>

<52>   </element>

<53>   <choice>

<54>     <element name="CoordinateInterpolatorEncodingParameter" minOccurs="0" maxOccurs="1">

<55>       <complexType>

<56>         <attribute name="keyQbits"   "type="xmta:numOfKeyQBits" use="optional" default="8"/>

<57>         <attribute  name="keyValueQBits" type="xmta:numOfKeyValueQBits use="optional"

default="16"/>

<58>         <attribute  name="transpose" type="xmta:transposeType" use="optional" default="&

quot;ON&quot;"/>
```

```
<59>      <attribute name="linearKeycoder" type="xmta:linearKeycoderType" use="optional"
      default="&quot;LINEAR&quot;"/>

<60>    </complexType>

<61>  </element>

<62>  <element name="IndexedFaceSetEncodingParameter" minOccurs="0" maxOccurs="1">

<63>    <complexType>

<64>      <attribute name="coordQBits" type="xmta:numOfCoordQBits" use="optional"
      default="10"/>

<65>      <attribute name="normalQBits" type="xmta:numOfNormalQBits" use="optional"
      default="9"/>

<66>      <attribute name="colorQBits" type="xmta:numOfColorQBits" use="optional"
      default="6"/>

<67>      <attribute name="texCoordQBits" type="xmta:numOfTexCoordQBits" use="optional"
      default="10"/>

<68>      <attribute name="coordPredMode" type="xmta:coordPredType" use="optional"
      default="2"/>

<69>      <attribute name="normalPredMode" type="xmta:normalPredType" use="optional"
      default="0"/>

<70>      <attribute name="colorPredMode" type="xmta:colorPredType" use="optional"
      default="0"/>
```

```

<71>      <attribute name="texCoordPredMode" type="xmta:texCoordPredType" use="optional"
        default="0"/>

<72>      <attribute name="errorResilience" type="xmta:errorResilienceType" use="optional"
        default="&quot;OFF&quot;"/>

<73>      <attribute name="bitsPerPacket" type="xmta:SInt32" use="optional"

<74> default="360"/>

<75>      <attribute name="boundaryPrediction" type="xmta:boundaryPredictionType" use="
        optional" default="0"/>

<76>      </complexType>

<77>      </element>

<78>      <element name="OrientationInterpolatorEncoding Parameter" minOccurs="0" maxOccurs="1">

<79>          <complexType>

<80>              <attribute name="keyQBits" type="xmta:numOfKeyQBits" use="optional" default="8"/>

<81>              <attribute name="keyValueQBits" type="xmta:numOfKeyValueQBits" use="optional"
                default="16"/>

<82>              <attribute name="preservingMode" type="xmta:preservingType" use="optional" default
                = "&quot; KEY&quot;"/>

<83>              <attribute name="dpcmMode" type = "xmta:orientationDpcmType" use="optional"
                default="0"/>

```

```
<84>      <attribute name="aacMode_X" type="xmta:aacType" use="optional" default="&quot;BINARY&quot;"/>

<85>      <attribute name="aacMode_Y" type="xmta:aacType" use="optional" default="&quot;BINARY&quot;"/>

<86>      <attribute name="aacMode_Z" type="xmta:aacType" use="optional" default="&quot;BINARY&quot;"/>

<87>      <attribute name="linearKeycoder" type="xmta:linearKeycoderType" use="optional"
default= "&quot; LINEAR&quot;"/>

<88>      </complexType>

<89>      </element>

<90>      <element name= "PositionInterpolatorEncoding Parameter" minOccurs="0" maxOccurs="1">

<91>          <complexType>

<92>              <attribute name="keyQBits" type="xmta:numOfKeyQBits" use="optional" default="8"/>

<93>              <attribute name="keyValueQBits" type="xmta:numOfKeyValueQBits" use="optional"
default="16"/>

<94>              <attribute name="preservingMode" type="xmta:preservingType" use="optional" default
= "&quot; KEY&quot;"/>

<95>              <attribute name="dpcmMode_X" type="xmta:positionDpcmType" use="optional"
default="0"/>
```

```
<96>      <attribute name="dpcmMode_Y" type="xmta:positionDpcmType" use="optional"
      default="0"/>

<97>      <attribute name="dpcmMode_Z" type="xmta:positionDpcmType" use="optional"
      default="0"/>

<98>      <attribute name="aacMode_X" type="xmta:aacType" use="optional" default="&
      quot;BINARY&quot;"/>

<99>      <attribute name="aacMode_Y" type="xmta:aacType" use="optional" default="&
      quot;BINARY&quot;"/>

<100>     <attribute name="aacMode_Z" type="xmta:aacType" use="optional" default="&
      quot;BINARY&quot;"/>

<101>     <attribute name="linearKeycoder" type="xmta:linearKeycoderType" use="optional"
      default = "&quot; LINEAR&quot;"/>

<102>     <attribute name="intra_X" type="xmta:intraType" use="optional" default="0"/>

<103>     <attribute name="intra_Y" type="xmta:intraType" use="optional" default="0"/>

<104>     <attribute name="intra_Z" type="xmta:intraType" use="optional" default="0"/>

<105>     </complexType>

<106>     </element>

<107>     <element name = "WaveletSubdivisionSurfaceEncoding Parameter" >

<108>     </element>

<109>     <element name="MeshGridEncodingParameter" >
```

<110> </element>

<111> </choice>

<112> <attribute name="type" type="xmta:SFInt32" use="optional" default="0"/>

<113> <attribute name="buffer" type="xmta:S FString" use="optional" />

<114> <attribute name="url" type="xmta:M FUrl" use="optional" />

<115> <attributeGroup ref="xmta:DefUseGroup"/>

<116> <complexType>

<117> </element>

<118> </schema>

<119> 1.2.2. Semantics

<120> BitWrapper 노드는 노드압축 스킴(scheme) 전용이다. 상기 압축된 표현은 BIFS 스트림 내에서 전송되거나 밖의 분리된 스트림으로 전송된다. 스트림이 BIFS 업데이트 내에서 전송될 때, "buffer" 필드는 압축된 표현을 포함한다. 스트림이 BIFS 업데이트 밖의 분리된 스트림으로 전송될 때, "url" 필드는 스트림의 url을 포함한다."buffer" 필드와 "url" 필드는 서로 배타적으로 사용된다. 즉 "buffer" 필드가 사용되면 "url" 필드는 사용되지 않고, "url" 필드가 사용되면 "buffer" 필드는 사용되지 않는다. "node" 필드는 압축된 표현을 갖는 노드를 포함한다. BitWrapper 노드는 "node"의 위치에서 사용될 수 있다. "type" 필드는 노드 압축 스킴이 사용되어야 한다는 것을 나타낸다. 0 이 "type" 필드 값의 디폴트 값이다. "type" 필드 값은 미래의 노드 압축 스킴이 개발될 것을 고려해서 나타낸 것이다. 이것에 대해서 디폴트 스킴을 정의한다.

- <121> "CoordinateInterpolatorEncodingParameter" 필드는 객체의 모든 정점좌표에 대한 키프레임 애니메이션 데이터(Coordinate Interpolator 노드 데이터)를 압축하기 위해 사용되는 매개인자를 포함한다. 이 경우 "node" 필드는 CoordinateInterpolator 노드이다.
- <122> "IndexedFaceSetEncodingParameter" 필드는 3차원 메쉬 정보(IndexedFaceSet 노드 데이터)를 압축하는데 사용되는 매개인자를 포함한다. 이 경우에 "node" 필드는 IndexedFaceSet 노드이다.
- <123> "OrientationInterpolatorEncodingParameter" 필드는 회전이동 키프레임 애니메이션 데이터(Orientation Interpolator 노드 데이터)를 압축하기 위해 사용되는 매개인자를 포함한다. 이 경우에 "node" 필드는 OrientationInterpolator 노드이다.
- <124> "PositionInterpolatorEncodingParameter" 필드는 위치이동 키프레임 애니메이션 데이터(PositionInterpolator 노드 데이터)를 압축하기 위해 사용되는 매개인자를 포함한다. 이 경우에 "node" 필드는 PositionInterpolator 노드이다. 매개인자를 사용하지 않을 때 매개인자가 파일 안에 명시되어야 하는 것은 아니다. 매개인자를 사용하여 압축이 실행될 때 매개인자 필드들은 배타적으로 사용되어야 한다. 그 이유는 "node" 필드가 한 번에 압축될 노드 데이터의 한 가지 유형(type)만 포함하기 때문이다. 또한 매개인자 필드들이 "<choice>" 엘리먼트로 그룹핑(grouping) 되었기 때문이다. 각 파라미터 필드에서 "attribute"는 다음 절에서 설명한다.
- <125> 압축된 표현이 분리된 스트림에 포함되어 전송될 때 노드 디코더는 어떤 틀에 맞춰 형성되어야 한다.

<126> 객체 기술자 스트림(object descriptor stream)에서 노드 디코더는 streamType 0x03과 objectTypeIndication 0x05에 대해 DecoderConfig descriptor에 나타나야 한다. 디코더는 AFXConfig descriptor로 형성된다.

<127> 1.3. *numOfKeyQBits*

<128> 1.3.1 Syntax

<129> <simpleType name="numOfKeyQBits">

<130> <restriction base="int">

<131> <minInclusive value="0">

<132> <maxInclusive value="31"/>

<133> </restriction>

<134> </simpleType>

<135> 1.3.2 Semantics

<136> numOfKeyQBits는 key 데이터의 양자화 비트 크기(quantization bit size)를 나타낸다.

이것은 정수(integer) 타입이다. numOfKeyQBits의 최소 값은 0 이고 최대 값은 31 이다.

<137> 1.4 numOfKeyValueQBits

<138> 1.4.1 Syntax

<139> <simpleType name="numOfKeyValueQBits">

<140> <restriction base="int"> <minInclusive value="0"/>

<141> <maxInclusive value="31"/>

<142> </restriction>

<143>       </simpleType>

<144>       1.4.2 Semantics

<145>       numOfKeyValueQBits는 keyValue 데이터의 양자화 비트 크기(quantization bit size)를 나타낸다. 이것은 integer 타입이다. numOfKeyQBits의 최소 값은 0이고 최대 값은 31이다.

<146>       1.5 *linearKeycoderType* 1.5.1 Syntax

<147>       <simpleType name="linearKeycoderType">

<148>       <restriction base="string">

<149>       <enumeration value="&quot;LINEAR&quot;"/>

<150>       <enumeration value="&quot;NOLINEAR&quot;"/>

<151>       </restriction>

<152>       </simpleType>

<153>       1.5.2 Semantics

<154>       linearKeycoderType은 string 타입입니다. linearKeycoderType은 linear key coder가 사용되는지 사용되지 않는지를 나타낸다.

<155>       1.6 *preservingType*

<156>       1.6.1 Syntax

<157>       <simpleType name="preservingType">

<158>       <restriction base="string">

<159>       <enumeration value="&quot;KEY&quot;"/>

<160>        <enumeration value="&quot;PATH&quot;"/>

<161>        </restriction></simpleType>

<162>        1.6.2 Semantics

<163>        preservingType은 string 타입이다. preservingType은 현재 모드가 key preserving 모드인지 path preserving 모드인지 나타낸다. .

<164>        1.7 *aacType*

<165>        1.7.1 Syntax

<166>        <simpleType name="aacType">

<167>        <restriction base="string">

<168>        <enumeration value="&quot;BINARY&quot;"/> <enumeration value="&quot;UNARY&quot;"/>

<169>        </restriction>

<170>        </simpleType>

<171>        1.7.2 Semantics

<172>        aacType은 string 타입이다. aacType은 각 keyValue 컴포넌트(X, Y, Z)에 대해 현재 모드가 BinaryAAC 모드인지 UnaryAAC 모드인지를 나타낸다.

<173>        1.8 *orientationDpcmType*

<174>        1.8.1 Syntax

<175>        <simpleType name="orientationDpcmType"> <restriction base="int">

<176>        <enumeration value="1"/>

<177>        <enumeration value="2"/>

<178>        </restriction>

<179>        </simpleType>

<180>        1.8.2 Semantics

<181>        orientationDpcmType이 각 keyValue 컴포넌트(X, Y, Z)에 대해 사용된 DPCM 차수를 나타낸다. 이것은 정수 타입이다. 만약 DPCM 차수가 "1"이라면 플래그가 0으로 셋팅된다. DPCM 차수가 "2"라면 1로 셋팅된다.

<182>        1.9 positionDpcmType

<183>        1.9.1 Syntax

<184>        <simpleType name="positionDpcmType">

<185>        <restriction base="int">

<186>        <enumeration value="0"/>

<187>        <enumeration value="1"/>

<188>        <enumeration value="2"/>

<189>        </restriction>

<190>        </simpleType>

<191>        1.9.2 Semantics

<192>        positionDpcmType은 각 keyValue 컴포넌트 (X, Y, Z)에 대해 사용된 DPCM 차수를 나타낸다. 이것은 정수 타입이다. DPCM 차수가 "1"이라면 플래그는 0으로 셋팅된다. DPCM 차수가 "2"이라면 1로 셋팅된다. SAD가 사용되면 2로 셋팅된다. SAD는 [2]에 자세히 설명되어 있다.

<193>        *1.10 intraType*

<194>        1.10.1 Syntax

<195>        <simpleType name="intraType">

<196>            <restriction base="int">     <enumeration value="0"/>

<197>                <enumeration value="1"/>

<198>            </restriction>

<199>        </simpleType>

<200>        1.10.2 Semantics

<201>        intraType은 Position Interpolator Compression에 사용된다. intraType은 각 keyValue  
컴포넌트 (X,Y,Z)에 대해 intra coding 모드가 사용되었는지 또는 사용되지 않았는지를 나타낸  
다.

<202>        *1.11 transposeType*

<203>        1.11.1 Syntax

<204>        <simpleType name="transposeType">

<205>            <restriction base="string">

<206>                <enumeration value="&quot;ON&quot;"/>

<207>                <enumeration value="&quot;OFF&quot;"/>

<208>            </restriction>

<209>        </simpleType>

<210> 1.11.2 Semantics

<211> transposeType은 transpose 모드 또는 vertex 모드에 대한 플래그이다. 만약, value가 "ON"으로 셋팅되었다면, transpose 모드가 사용되고, 그렇지 않으면, vertex 모드가 사용된다.

<212> 1.12 numOfCoordQBits

<213> 1.12.1 Syntax

<214> <simpleType name="numOfCoordQBits">

<215> <restriction base="int">

<216> <minInclusive value="1"/>

<217> <maxInclusive value="24"/>

<218> </restriction>

<219> </simpleType>

<220> 1.12.2 Semantics

<221> numOfCoordQBits는 geometry에 대한 양자화 스텝(quantization step)을 나타낸다.

numOfCoordQBits의 최소 값이 1이고 최대 값은 24이다.

<222> 1.13 numOfNormalQBits

<223> 1.13.1 Syntax

<224> <simpleType name="numOfNormalQBits">

<225> <restriction base="int">

<226> <minInclusive value="3"/> <maxInclusive value="31"/>

<227>       </restriction>

<228>       </simpleType>

<229>       1.13.2 Semantics

<230>       numOfNormalQBits는 normal에 대한 양자화 스텝(quantization step)을 나타낸다.

numOfNormalQBits의 최소 값이 3이고 최대 값은 31이다.

<231>       1.14 numOfColorQBits

<232>       1.14.1 Syntax

<233>       <simpleType name="numOfColorQBits"> <restriction base="int">

<234>       <minInclusive value="1"/>

<235>       <maxInclusive value="16"/>

<236>       </restriction>

<237>       </simpleType>

<238>       1.14.2 Semantics

<239>       numOfColorQBits는 color에 대한 양자화 스텝(quantization step)을 나타낸다.

numOfColorQBits의 최소 값이 1이고 최대 값은 16이다.

<240>       1.15 numOfTexCoordQBits

<241>       1.15.1 Syntax

<242>       <simpleType name="numOfTexCoordQBits">

<243>       <restriction base="int">

<244>       <minInclusive value="1"/>

<245>        <maxInclusive value="16"/>

<246>        </restriction>

<247>        </simpleType>

<248>        1.15.2 Semantics

<249>        NumOfTexCoordQBits는 texture coordinates에 대한 양자화 스텝(quantization step)을 나타낸다. NumOfTexCoordQBits의 최소 값이 1이고 최대 값은 16이다.

<250>        *1.16 coordPredType*

<251>        1.16.1 Syntax

<252>        <simpleType name="coordPredType">

<253>        <restriction base="int">

<254>        <enumeration value="0"/>

<255>        <enumeration value="2"/>

<256>        </restriction>

<257>        </simpleType>

<258>        1.16.2 Semantics

<259>        CoordPredType은 메쉬의 vertex coordinates를 재구성하기 위해 사용되는 prediction type이다. no\_prediction이 사용되면, CoordPredType 값은 0으로 셋팅된다.  
parallelogram\_prediction이 사용되면, 2로 셋팅된다.

<260>        *1.17 normalPredType*

<261>        1.17.1 Syntax

<262> <simpleType name="normalPredType">

<263> <restriction base="int">

<264> <enumeration value="0"/>

<265> <enumeration value="1"/>

<266> <enumeration value="2"/>

<267> </restriction>

<268> </simpleType>

<269> 1.17.2 Semantics

<270> NormalPredType은 어떻게 normal values가 예측(predict)될 수 있는지 나타낸다.

no\_prediction이 사용되면, 값이 0으로 셋팅되고, tree\_prediction이 사용되면, 1로 셋팅되고, parallelogram\_prediction이 사용되면, 2로 셋팅된다.

<271> 1.18 colorPredType

<272> 1.18.1 Syntax

<273> <simpleType name="colorPredType">

<274> <restriction base="int">

<275> <enumeration value="0"/>

<276> <enumeration value="1"/>

<277> <enumeration value="2"/>

<278> </restriction>

<279> </simpleType>

<280> 1.18.2 Semantics

<281> colorPredType은 어떻게 colors가 예측(predict)될 수 있는지 나타낸다. no\_prediction이 사용되면, 값은 0이 되고, tree\_prediction이 사용되면, 1이 셋팅되고, parallelogram\_prediction이 사용되면, 2로 셋팅된다.

<282> 1.19 texCoordPredType

<283> 1.19.1 Syntax

<284> <simpleType name="texCoordPredType">

<285> <restriction base="int"> <enumeration value="0"/>

<286> <enumeration value="2"/>

<287> </restriction>

<288> </simpleType>

<289> 1.19.2 Semantics

<290> texCoordPredType은 어떻게 colors가 예측(predict)될 수 있는지 나타낸다.

no\_prediction이 사용되면, 값은 0이 되고, parallelogram\_prediction이 사용되면, 2로 셋팅된다.

<291> 1.20 errorResilienceType

<292> 1.20.1 Syntax

<293> <simpleType name="errorResilienceType">

<294> <restriction base="string">

<295> <enumeration value="&quot;ON&quot;"/>

<296> <enumeration value="&quot;OFF&quot;"/>

<297> </restriction>

<298> </simpleType>

<299> 1.20.2 Semantics

<300> ErrorResilienceType은 오류 강인성(Error Resilience) 모드가 사용되었는지 나타낸다.

오류 강인성이 사용되지 않으면 "OFF"로 셋팅되고, 오류 강인성이 사용되면 "ON"으로 셋팅된다

. 만약 ""ON"으로 된 경우에만 boundaryPredictionType과 bitsPerPacket이 사용될 수 있다.

<301> 1.21 boundaryPredictionType

<302> 1.21.1 Syntax

<303> <simpleType name="boundaryPredictionType">

<304> <restriction base="int"> <enumeration value="0"/>

<305> <enumeration value="1"/>

<306> </restriction>

<307> </simpleType>

<308> 1.21.2 Semantics

<309> BoundaryPredictionType은 boundary prediction의 타입을 나타낸다. 값이 0이라면 제한

된 예측(restricted prediction)이 사용되고, 값 1이 사용된다면 확장 예측(extended

prediction)이 사용된다.

<310> 1.22 bitsPerPacket

<311> 1.22.1 Syntax

<312> bitsPerPacket의 구문은 SFInt32 타입과 똑같다.

<313> <simpleType name="SFInt32">

<314> <restriction base="int"/>

<315> </simpleType>

<316> 1.22.2 Semantics

<317> BitsPerPacket은 에러 강인 비스트림(error resilient bitstream)에 대한 패킷 크기를 나타낸다. 이 값은 에러 강인 모드에서 각 부분의 크기를 결정한다. bitsPerPacket의 타입은 SFInt32이다. 디폴트 값은 360이다.

<318> 2. BitWrapperEncodingHints에 대한 XMT-A 스키마

<319> 2.1. BitWrapperEncodingHints

<320> 2.1.1. Syntax

<321> 다음은 BitWrapperEncodingHints의 구문이다.

<322> <!-- Declaration of BitWrapperEncodingHints -->

<323> <element name="StreamSource">

<324> <complexType> <choice minOccurs="0" maxOccurs="unbounded">

<325> <element ref="xmta:BitWrapperEncodingHints"/>

<326> </choice>

<327> ...

<328> </complexType>

```
<329>          </element>

<330>      ...

<331>      <!-- Definition of BitWrapperEncodingHints -->

<332>          <element name="BitWrapperEncodingHints">

<333>              <complexType>

<334>                  <element name="BitWrapper3DMCEncodingHints">

<335>                      <complexType>

<336>                          <sequence>

<337>                              <element name="sourceFormat">

<338>                                  <complexType>

<339>                                      <sequence>

<340>                                          <element ref="xmta:param" minOccurs="0" maxOccurs="
unbounded"/>

<341>                                              </sequence>

<342>                                      </complexType>

<343>                              </element>

<344>                      <element name="targetFormat">

<345>                          <complexType>

<346>                              <sequence>
```

```
<347>                <element ref="xmta:param" minOccurs="0" maxOccurs="
unbounded"/>

<348>                </sequence>

<349>            </complexType>

<350>        </element>

<351>    </sequence>

<352>    </complexType>

<353>    </element>

<354>    <element name="BitWrapperICEncodingHints">

<355>        <complexType>

<356>            <sequence>

<357>                <element name="sourceFormat">

<358>                    <complexType>

<359>                        <sequence>

<360>                            <element ref="xmta:param" minOccurs="0" maxOccurs="unbounded"/>

<361>                        </sequence>

<362>                    </complexType>

<363>                </element>

<364>                <element name="targetFormat">
```

```

<365>                <complexType>

<366>                <sequence>

<367>                <element ref="xmta:param" minOccurs="0" maxOccurs="
unbounded"/>

<368>                </sequence>

<369>                </complexType>

<370>                </element>

<371>                </sequence>

<372>            </complexType>

<373>            </element>

<374>            <element name="OthersEncodingHints">

<375>            ...

<376>            </element>

<377>        </complexType>

<378>    </element>

<379>    2.1.2. Semantics

<380>    BitWrapperEncodingHints는 스크립트(.mux) 파일에서 "MuxInfo description을 명세하기
    위해 사용된다. 그 결과 binary textual format과 일치한다. BitWrapper 노드가 그 안에 있는
    "url" 필드를 사용하는 아웃-밴드(out-band) 시나리오에 대해 사용된다.

    BitWrapperEncodingHints는 스트림 포맷의 타입에 따른
  
```

"MuxInfo" description에 사용되는 적절한 정보를 설명하고 있다.3. XMT2MUX 스타일 쉬트에 BitWrapperEncodingHints에 대한 수정

<381>       다음은 XMT2MUX 스타일 쉬트에 MuxInfo 와 BitWrapperEncodingHints 구문을 다음과 같이 수정하였다.

<382>       3.1 Syntax

<383>       원래의 구문은 다음과 같다.

<384>       <xsl:template match="xmt:StreamSource"> muxInfo {

<385>       fileName <xsl:value-of select="@url"/><xsl:text>

<386>       <!-- what to do for urls? --></xsl:text>

<387>       <!-- if no encoding hints are given, it is assumed the stream is of type BIFS.

otherwise

<388>       the streamFormat should be declared in the parameter element of sourceFormat or targetFormat

<389>       (name 'streamFormat' and value corresponding to a streamFormat recognised by BIFSEnc) -->

<390>       <xsl:if test="not(xmt:EncodingHints)">streamFormat BIFS<xsl:text>

<391>       </xsl:text></xsl:if>

<392>       <xsl:apply-templates select="xmt:EncodingHints|xmt:BIFSEncodingHints| xmt:FBAEncodingHints"/>

<393>       }

```

<394>      </xsl:template>

<395>      수정된 구문은 다음과 같다.

<396>      <xsl:template match="xmt:StreamSource"> muxInfo MuxInfo{

<397>      fileName <xsl:value-of select="@url"/>

<398>      <xsl:apply-templates select="xmt:EncodingHints|xmt:BIFSEncodingHints|

xmt:FBAEncodingHints|xmt:BitWrapperEncodingHints"/>

<399>      <xsl:if test="not(xmt:EncodingHints|xmt:BitWrapperEncodingHints)"> streamFormat

BIFS<xsl:text></xsl:text></xsl:if>

<400>      <xsl:if test="xmt:BitWrapperEncodingHints"><xsl:text>

<401>      </xsl:text></xsl:if>

<402>      <xsl:apply-templates select="xmt:BitWrapper3DMCEncodingHints|xmt:

BitWrapperICEncodingHints| xmt:OthersEncodingHints"/>

<403>      }

<404>      <xsl:template match="xmt:BitWrapperEncodingHints">

<405>      <xsl:apply-templates select="xmt:BitWrapper3DMCEncodingHints|xmt:

BitWrapperICEncodingHints|xmt:OthersEncodingHints"/>

<406>      <xsl:apply-templates select="xmt:sourceFormat|xmt:targetFormat"/>

<407>      </xsl:template>

<408>      <xsl:template match="xmt:BitWrapper3DMCEncodingHints">

```

```
<409>      <xsl:apply-templates select="xmt:sourceFormat|xmt:targetFormat"/>

<410>      streamFormat 3DMC<xsl:text>

<411>      </xsl:text>

<412>      </xsl:template>

<413>      <xsl:template match="xmt:BitWrapperICEncodingHints">

<414>      <xsl:apply-templates select="xmt:sourceFormat|xmt:targetFormat"/>

<415>      streamFormat InterpolatorCompression<xsl:text>

<416>      </xsl:text>

<417>      </xsl:template>

<418>      <xsl:template match="xmt:OthersEncodingHints">

<419>      <xsl:apply-templates select="xmt:sourceFormat|xmt:targetFormat"/>

<420>      streamFormat BIFS<xsl:text>

<421>      </xsl:text>

<422>      </xsl:template>

<423>      <xsl:template match="xmt:sourceFormat">

<424>      <xsl:apply-templates select="xmt:param"/>

<425>      </xsl:template>

<426>      <xsl:template match="xmt:targetFormat">

<427>      <xsl:apply-templates select="xmt:param"/>
```

<428> </xsl:template>

<429> </xsl:template>

<430> 3.2 semantics

<431> 기존 구문에는 MP4 부호화기로 전송되는 비트스트림에 대한 mux 정보(어떤 이름의 비트스트림인지, 어떤 종류의 비트스트림인지 예: A.bifs, A.od)가 XMT2MUX 스타일 쉬트에 제대로 기술(description)되지 않았다. 또한 BitWrapper에서 URL을 사용할 경우, URL을 통해 전송되는 비트스트림에 대한 정보(파일이름 실시예 3과 같이 "bunny-15000-tcp.m3d", 비트스트림의 종류)에 대해서 XMT2MUX 스타일 쉬트에 없었다. 따라서 BitWrapper에서 제공되는 비트스트림을 실어나르는 파일이름과 비트스트림의 종류를 기술(description)하였다. 구체적으로 3차원 애니메이션 데이터 압축에 대해서는 InterpolatorCompression streamformat, 3D 메쉬 데이터 압축에 대해서는 3DMC streamformat을 기술(description)하였다.

<432> 4. ObjectDescriptorUpdate에 대한 수정

<433> 4.1 syntax

<434> 다음은 XMT2BIFS 스타일 쉬트에서 ObjectDescriptorUpdate 구문을 다음과 같이 수정하였다. 원래 구문은 다음과 같다.

<435> <xsl:template match="xmt:ObjectDescriptorUpdate">

<436> UPDATE OD [

<437> <xsl:apply-templates select="xmt:OD"/>

<438> ]

<439> </xsl:template>

<440> 수정된 구문은 다음과 같다.

<441> <xsl:template match="xmt:ObjectDescriptorUpdate"> UPDATE OD [

<442> <xsl:apply-templates select="xmt:OD"/>

<443> ]

<444> </xsl:template>

<445> <xsl:template match="xmt:OD">

<446> <xsl:apply-templates select="xmt:ObjectDescriptorBase" />

<447> </xsl:template>

<448> <xsl:template match="xmt:ObjectDescriptorBase">

<449> <xsl:apply-templates select="xmt:ObjectDescriptor|xmt: InitialObjectDescriptor"/>

<450> </xsl:template>

<451> <xsl:template match="xmt:ObjectDescriptor">

<452> ObjectDescriptor {

<453> <xsl:if test="@objectDescriptorID"> objectDescriptorID <xsl:value-of select="

@objectDescriptorID"/></xsl:if>

<454> <xsl:text></xsl:text><xsl:apply-templates select="xmt:URL" />

<455> }

<456> </xsl:template>

<457> <xsl:template match="xmt:URL">

<458> <xsl:if test="@URLstring"> muxScript

<459> <xsl:value-of select="@URLstring"/></xsl:if>

<460> <xsl:text></xsl:text>

<461> </xsl:template>

<462> 4.2 Semantics

<463> 기존 구문에는 "Update OD 내부에 어떤 기술(description)도 없었다. "Update OD 는 저작자가 "url" 필드를 통해서 다른 엘리먼트 스트림과 연결하고자 할 경우, 장면 설명 스트림(scene description stream (BIFS stream))을 링크할 때 사용된다. 또한 Update OD 내부에 ObjectDescriptorID와 스크립트 파일 이름을 설명하는 부분도 추가하였다. 이것은 binary textual format과 일치한다. 본 발명의 동작을 설명하기로 한다. (1) 원본 데이터를 매개인자(encoding parameter)를 사용해 비트스트림을 만들어 버퍼로 전송하는 경우이다. 도 2와 도 4를 참조하여 원본 데이터를 매개 인자를 사용해 비트스트림을 만들어 버퍼로 전송하는 실시 예를 설명하기로 한다.

<464> 도 4는 물체 A(예 : 컵)에 대한 3차원 데이터(기하 정보, 연결정보, 색상정보 등)를 매개인자를 사용해 압축하여 물체 A에 압축된 비트스트림(bitstream)을 버퍼를 사용해 "BufferWithEP.m3d"로 전송하는 방법을 나타낸다. 도 2에 도시된 바와 같이 XMT 파일이 입력으로 들어오면, XMT 파서에서 XMT-A 스키마를 참조하고, 스타일 쉬트(XMT2BIFS, XMT2MUX)를 사용하여 비트스트림을 장면 데이터와 함께 .scene에 대한 비트스트림에 실어 전송한다. 이 때, .mux 내의 정보에는 MPEG-4 Encoder인 BIFS Encoder가 실행된 결과인 .bifs/.od 가 포함된다. 그리고 .mux 내의 정보와 .bifs/.od 파일이 MP4 Encoder의 입력으로 들어와 최종 결과 파일

(.mp4)이 생성되고, MPEG-4 플레이어(Player)에서 그 결과를 볼 수 있다. 실시 예를 들어 보다 자세한 설명을 한다.[실시예 1] 다음은 BitWrapper를 사용해 원본의 3차원 메쉬 정보를 매개 인자를 사용해 비트스트림을 만들고 버퍼를 사용해 전송하는 XMT 예제이다.

```

<465> <Header>

<466>   <InitialObjectDescriptor objectDescriptorID="1" binaryID="1" >       <Descr>

<467>       <esDescr>

<468>   <ES_Descriptor ES_ID="xyz" binaryID="201" OCR_ES_ID="xyz">

<469>       ...

<470>       <StreamSource>

<471>           <BIFSEncodingHints>

<472>       <sourceFormat>

<473>           <param value=" BufferWithEP.bif"> </param>

<474>       </sourceFormat>

<475>           </BIFSEncodingHints>

<476>       </StreamSource>

<477>   </ES_Descriptor>

<478>       </esDescr>

<479>       </Descr>

<480> </InitialObjectDescriptor>

```

```
<481> </Header>

<482> ...

<483> <BitWrapper type="0" buffer="MyIndexFaceSet.m3d">

<484>     <node>

<485>         <IndexedFaceSet ccw="TRUE" solid="TRUE"

<486>             coordIndex="0, 1, 2, -1, 3, 4, 5, -1,..." normalPerVertex="TRUE">

<487>                 <coord DEF="Box-COORD"></coord>

<488>                 <Coordinate point= "0 1 2, 0 2 3, 4 0 1, 1 5 4, 5 1 2, 2 6 5, ..."><

/Coordinate>

<489>                 <normal>

<490>                     <Normal vector="0.7859 -0.341 -0.5157, 0.3812 -0.801 0.4615, ...">

<491>                         </Normal></normal>

<492>                     </IndexedFaceSet>

<493>                 </node>

<494>         <IndexedFaceSetEncodingParameter coordQBits="10 normalQBits="9"

coordPredMode="2" normalPredMode="0"

errorResilience="OFF"> </IndexedFaceSetEncodingParameter>

<495> </BitWrapper>

<496> ...
```

<497>       도 2에서 실시예 1과 같이 BitWrapper 노드를 사용해 3차원 메쉬정보와 그 매개 인자를 사용해 압축하는 XMT 표현의 입력 파일이 들어오면, XMTRef 파서(Parser)에서 BitWrapper 노드 안에서의 3차원 메쉬정보 매개 인자를 사용해 압축하여 비트스트림을 만들어 낸다. 이것은 BitWrapper 노드와 매개 인자에 대한 XMT-A 스키마 정의를 따른다. 또한 XMT-A 스키마와 스타일 쉬트(XMT2BIFS와 XMT2MUX) 파일을 사용해, MPEG-4 부호화기의 입력 파일(".scene" 파일과 ".mux" 파일)을 만들어 낸다. 이 결과는 다음과 같다.

<498>       - "BufferWithEP.scene" 파일 -

<499>       ..

<500>       BitWrapper {

<501>       node IndexedFaceSet {

<502>           ccw="TRUE"

<503>           solid="TRUE"

<504>           coordIndex="0, 1, 2, -1, 3, 4, 5, -1, ..."

<505>           normalPerVertex="TRUE"

<506>       coord DEF Box-COORD Coordinate {       point [0 1 2, 0 2 3, 4 0 1, 1 5 4, 5 1 2,  
2 6 5, ... }

<507>           }

<508>       normal Normal {

<509>           Vector[0.7859, -0.341, -0.5157, 0.3812, -0.801, 0.4615, ...]

<510>           }

```
<511>      }

<512>      IndexedFaceSetEncodingParameter{

<513>          coordQBits  10

<514>          NormalQBits  9

<515>          coordPredMode  2

<516>          normalPredMode  0

<517>          errorResilience  OFF

<518>      }

<519>      type 0

<520>          buffer "MyIndexFaceSet.m3d"

<521>  }

<522>  ..

<523>  - "BufferWithEP.mux" 파일 -InitialObjectDescriptor {

<524>  ...

<525>      muxInfo MuxInfo{

<526>          fileName  "BufferWithEP.bif"

<527>          streamFormat BIFS

<528>      }

<529>  }
```

<530> 위와 같은 .scene 파일과 .mux 파일은 MPEG-4 플레이어의 입력으로 사용되어 BIFS 부호화기를 거쳐 bifs/od 파일을 생성하고, 이 bifs/od 파일과 mux 파일이 MP4 부호화기에 입력으로 들어가 하나의 비트스트림(".mp4") 파일을 생성한다. 상기 ".mp4" 파일은 최종 결과로서 MPEG-4 플레이어에서 시각화된다.

<531> (2) 이미 압축된 비트스트림을 사용해 버퍼(buffer)로 전송하는 경우이다. 도 2와 도 5를 참조하여 이미 압축된 비트스트림을 사용해 버퍼(buffer)로 전송하는 실시 예를 설명하기로 한다. 도 5의 내용은 다음과 같다. 물체 A(예 : 컵)에 대한 3차원 데이터(기하 정보, 연결정보, 색상정보 등)가 이미 압축되어 물체 A에 압축된 비트스트림(bitstream) 형태로 사용된다. 이 비트스트림은 버퍼를 통해 "BufferWithoutEP .m3d"로 전송된다. 상기 "(1) 원본 데이터를 매개 인자를 사용해 비트스트림을 만들어 버퍼로 전송하는 경우" 와 차이점은 XMT 입력 파일에서 BitWrapper 노드에 물체 A에 대한 3차원 데이터와 매개 인자를 사용해 직접 압축하는 것이 아니라, 이미 압축된 물체 A에 대한 비트스트림을 버퍼로 전송한다는 점이다. 도 2에 도시된 바와 같이 XMT 파일(200)이 입력으로 들어오면, XMT 파서(210)에서 XMT-A 스키마(240)를 참조하고, XMT2BIFS 스타일시트(230)와 XMT2MUX 스타일시트(220)를 사용하여 이 압축한 비트스트림을 장면 데이터와 함께 .scene에 대한 비트스트림에 실어 전송한다. 이 때, .mux 내의 정보에는 MPEG-4 부호화기인 BIFS 부호화기(250)가 실행된 결과 .bifs/.od 가 포함된다. 그리고, .Mux 내의 정보와 .bifs/.od 파일이 MP4 부호화기(260)의 입력으로 들어와 최종 결과 파일(.mp4)이 생성되고, MPEG-4 플레이어에서 그 결과를 볼 수 있다. 실시 예를 들어 더 자세한 설명을 하겠다.

<532> [실시예 2] 다음은 BitWrapper를 사용해 이미 압축된 3차원 메쉬정보의 비트스트림을 버퍼를 사용해 전송하는 XMT 예제이다.

<533> ...  
<534> <Header>  
<535> <InitialObjectDescriptor objectDescriptorID="1" binaryID="1" > <Descr>  
<536> <esDescr>  
<537> <ES\_Descriptor ES\_ID="xyz" binaryID="201" OCR\_ES\_ID="xyz">  
<538> ...  
<539> <StreamSource>  
<540> <BIFSEncodingHints>  
<541> <sourceFormat>  
<542> <param value="BufferWithoutEP.bif"> </param>  
<543> </sourceFormat>  
<544> </BIFSEncodingHints>  
<545> </StreamSource>  
<546> </ES\_Descriptor>  
<547> </esDescr>  
<548> </Descr>  
<549> </InitialObjectDescriptor>

```
<550> </Header>

<551> <Body>

<552>   <Replace>

<553>   <Scene>

<554>   <Group>

<555>     <children>

<556>       <Shape>

<557>         <geometry DEF="MY_BOX">

<558>           <BitWrapper type="0"  buffer="MyIndexFaceSet.m3d">           <node>

<559>             <IndexedFaceSet>

<560>               <coord DEF="Box-COORD"></coord>

<561>               <Coordinate></Coordinate>

<562>             </node>

<563>           </BitWrapper>

<564>         </geometry>

<565>       </Shape>

<566>     </children>

<567>   </Group>
```

<568> </Scene>

<569> </Replace>

<570> </Body>

<571> ...

<572> 도 2에서 실시예 2와 같이 BitWrapper 노드를 사용해 이미 압축된 3차원 메쉬정보 비트스트림을 버퍼로 전송하는 XMT 표현의 입력 파일이 들어오면, XMTRef 파서(Parser)에서 BitWrapper 노드 안의 버퍼로 전송되는 압축된 3차원 메쉬 정보 비트스트림을 만들어 낸다. 이것은 BitWrapper 노드와 매개 인자에 대한 XMT-A 스키마 정의를 따른다. 또한 XMT-A 스키마와 스타일 쉬트(XMT2BIFS와 XMT2MUX) 파일을 사용해, MPEG-4 부호화기의 입력 파일(".scene" 파일과 ".mux" 파일)을 만들어 낸다. 이 결과는 다음과 같다.- BufferWithoutEP.scene 결과 -

<573> ...

<574> BitWrapper {

<575> node IndexedFaceSet {

<576> coord DEF Box-COORD Coordinate {

<577> }

<578> }

<579> type 0

<580> buffer "MyIndexFaceSet.m3d"}

<581> ...

<582> - BufferWithoutEP.mux 결과 -

```

<583>      InitialObjectDescriptor {
<584>      ...
<585>      muxInfo MuxInfo{
<586>          fileName   "BufferWithoutEP.bif"
<587>          streamFormat BIFS
<588>      }
<589>  }

```

<590> 위와 같은 .scene 파일과 .mux 파일은 MPEG-4 부호화기의 입력으로 사용되어, BIFS 부호화기를 거쳐 bifs/od 파일을 생성하고, 이 bifs/od 비트스트림과 mux 파일이 MP4 부호화기에 입력으로 들어가 하나의 비트스트림(".mp4") 파일을 생성한다. 이 "mp4" 파일은 최종 결과로서 MPEG-4 플레이어에서 시각화된다.

<591> 3) 원본 데이터를 매개 인자를 사용해 비트스트림을 만들어 URL로 전송하는 경우이다. 도 2와 도 6을 참조하여 원본 데이터를 매개 인자를 사용해 비트스트림을 만들어 URL로 전송하는 실시 예를 설명하겠다.

<592> 도 6은 BitWrapper 노드 안에서 물체 A(예 : 컵)에 대한 3차원 데이터(기하 정보, 연결 정보, 색상정보 등)를 매개인자를 사용해 압축하여 물체 A의 압축된 비트스트림(bitstream)을 URL을 사용해 전송하는 방법을 나타낸다. 여기서는 상술한 버퍼를 통해 전송하는 방법과 다른 점 2가지가 있다. 첫째, BitWrapper에서 url ID(예:12)를 사용하고, 상기 url ID 와 같은 ObjectDescriptorID를 갖는 BitWrapperEncodingHints에 물체 A의 압축된 비트스트림의 위치정보(파일 "bunny-15000-tcp .m3d")가 명시되고, 이를 사용해 압축된 비트스트림이 전송된다.

둘째, 물체 A의 압축된 비트스트림의 위치 정보는 .mux의 위치 정보에 포함된다. 이것은 BitWrapper 노드의 url ID와 같은 ObjectDescriptorUpdate에 명시된다. 상기 .mux의 위치정보는 물체 A의 압축된 비트스트림의 위치정보(파일 "bunny-15000-tcp.m3d")를 링크하고 있고, 물체 A의 압축된 비트스트림의 위치 정보에는 물체 A의 압축 비트스트림을 링크하고 있다. 따라서 url을 사용해 "bunny-15000-tcp.m3d" 파일의 비트스트림이 전송된다. 이러한 방법이 도 2에서와 같이 입력 XMT 파일에 명시되어 입력으로 들어오면, XMT 파서에서 XMT-A 스키마를 참조하고, 스타일 쉬트(XMT2BIFS, XMT2MUX)를 사용하여 .mux의 위치정보를 장면 데이터와 함께 .scene에 대한 비트스트림에 실어 전송한다. 그리고 .mux 내의 내용에는 MPEG-4 부호화기인 BIFS 부호화기가 실행된 결과인 .bifs/.od 가 포함된다. 또한 물체 A의 압축된 비트스트림의 위치정보("bunny-15000-tcp.m3d")도 포함된다.

<593> 이렇게 해서 물체 A의 압축된 비트스트림의 위치 정보(.mux 내의 내용)와 .bifs/.od 파일이 MP4 부호화기의 입력으로 들어와 하나의 비트스트림 파일(.mp4)이 생성되고, MPEG-4 플레이어에서 그 결과를 볼 수 있다. 실시 예를 들어 더 자세한 설명을 하겠다.

<594> [실시예 3] 다음은 BitWrapper를 사용해 원본의 3차원 메쉬정보를 매개 인자를 사용해 비트스트림을 만들고 URL을 사용해 전송하는 XMT 예제이다.

<595> <Header>

<596> ...

<597> <ObjectDescriptor objectDescriptorID="12" binaryID="12">

<598> <Descr>

<599> <esDescr>

<600> <ES\_Descriptor ES\_ID="PI" binaryID="211" OCR\_ES\_ID="PIC"> ...

<601> <StreamSource>

<602> <BitWrapperEncodingHints>

<603> <BitWrapper3DMCEncodingHints>

<604> <sourceFormat>

<605> <paramvalue="bunny-15000-tcp.m3d"> </param>

<606> </sourceFormat>

<607> </BitWrapper3DMCEncodingHints>

<608> </BitWrapperEncodingHints>

<609> </StreamSource>

<610> </ES\_Descriptor>

<611> </esDescr>

<612> </Descr>

<613> </ObjectDescriptor>

<614> </Header>

<615> <Body>

<616> <Replace>

<617> <Scene>

<618> <Group>

```
<619>    <children>

<620>    <Shape>

<621>        <geometry DEF="MY_BOX">

<622>    <Box size="69"></Box>

<623>    <BitWrapper type="0" url="12">        <node>

<624>        <IndexedFaceSet ccw="TRUE"

<625>            solid="TRUE"

<626>            coordIndex="0,1,2,-1,3,4,5,-1, ..."

<627>            normalPerVertex="TRUE">

<628>        <coord DEF="Box-COORD"></coord>

<629>    <Coordinate point="012,023,401,154,512,265, ..."></Coordinate>

<630>        <normal>

<631>            <Normal vector="0.7859 -0.341 -0.5159, 0.3821 -0.801 0.4615, ...">    </Normal>

<632>        </normal>

<633>    </IndexedFaceSet>

<634>    <node>

<635>        </IndexedFaceSetEncodingParameter coordQits="10"

<636>        normalQBits="9"

<637>        coordPreMode="2"
```

```
<638>          normalPredMode="0"

<639>          errorResilience="OFF">          </IndexedFaceSetEncodingParameter>

<640>          </BitWrapper>

<641>    </geometry>

<642>  </Shape>

<643>    </children>

<644>  </Group>

<645> </Scene>

<646> </Replace>

<647> <ObjectDescriptorUpdate>

<648>   <OD>

<649>     <ObjectDescriptorBase>

<650>       <ObjectDescriptor objectDescriptorID="12" binaryID="12">   <URL URLstring="
        UrlWithEP.scr"></URL>     </ObjectDescriptor>

<651>     </ObjectDescriptorBase>

<652>   </OD>

<653> </ObjectDescriptorUpdate>

<654> </Body>
```

<655> 도 2에서 실시예 3과 같이 BitWrapper 노드를 사용해 3차원 메쉬 정보와 그 매개 인자를 사용해 압축하는 XMT 표현의 입력 파일이 들어오면, XMTRef 파서(Parser)에서 BitWrapper 노드 안에서의 3차원 메쉬 정보 매개 인자를 사용해 압축하여 비트스트림을 만들어 낸다. 이것은 BitWrapper 노드와 매개 인자에 대한 XMT-A 스키마 정의를 따른다. 또한 XMT-A 스키마와 스타일 쉬트(XMT2BIFS와 XMT2MUX) 파일을 사용해, MPEG-4 부호화기의 입력 파일(".scene" 파일과 ".mux" 파일)을 만들어 낸다. 이 결과는 다음과 같다.- UrlWithEP.scene 결과 -

<656> ...

<657> BitWrapper {

<658>     node IndexedFaceSet {

<659>         ccw TRUE

<660>         coordIndex [ 0, 1, 2, -1, 3, 4, 5, -1, ... ]     normalPerVertex TRUE

<661>         solid TRUE

<662>     coord DEF Box-COORD Coordinate {

<663>         point [012, 023, 401, 154, 512, 265, ... ]

<664>         }

<665>     normal Normal {

<666>         vector [0.7859, -0.341, -0.5159, 0.3821, -0.801, 0.4651, ... ]

<667>     }

<668> }

<669>     IndexedFaceSetEncodingParameter {

```
<670>      coordQBits  10
<671>      normalQBits 9
<672>      coordPredMode  2
<673>      normalPredMode  0
<674>      errorResilience  OFF
<675>    }
<676>      type 0
<677>      url 12
<678>    }
<679>  ..
<680> UPDATE OD [
<681> ObjectDescriptor {   objectDescriptorID 12
<682>   muxScript UrlWithEP.scr
<683> }
<684> ]
<685> - "UrlWithEP.mux 결과 -
<686> ...
<687> ObjectDescriptor {
<688> objectDescriptorID  12
```

```
<689> esDescr [
<690>   ES_Descriptor {
<691>   ES_ID    211
<692>   ...
<693>   muxInfo MuxInfo{
<694>   fileName   "bunny-15000-tcp.m3d"
<695>   streamFormat 3DMC
<696>   }
<697> ]
<698> }
```

<699> 위와 같은 .scene 파일과 .mux 파일은 MPEG-4 부호화기의 입력으로 사용되어, BIFS 부호화기를 거쳐 .bifs/.od 파일을 생성하고, 이 .bifs/.od 비트스트림과 mux 파일이 MP4 부호화에 입력으로 들어가 비트스트림(".mp4") 파일을 생성한다. 이 "mp4" 파일은 최종 결과로서 MPEG-4 플레이어에서 시각화된다.

<700> (4) 이미 압축된 비트스트림을 사용해 URL로 전송하는 경우이다. 도 2와 도 6을 참조하여 이미 압축된 비트스트림을 사용해 URL로 전송하는 실시 예를 설명하기로 한다.

<701> 도 7은 BitWrapper 노드 안에서 이미 압축된 물체 A(예 : 컵)에 대한 3차원 데이터(기하 정보, 연결정보, 색상 등)의 비트스트림(bitstream)이 URL을 사용해 전송하는 방법을 나타낸다. 이 경우 버퍼와의 다른 점에 대해서는 "3) 원본 데이터를 매개 인자를 사용해 비트스트림을 만들어 URL로 전송하는 경우"와 같다. 그러나 직접 물체 A에 대한 3차원 데이터를 매개 인

자를 사용해서 압축하는 것이 아니라, 이미 압축된 물체 A에 대한 비트스트림을 URL을 사용해 전송하는 방법이 다르다. 따라서 도 2에서와 같이 입력 XMT 파일에 BitWrapper 노드에 원본 데이터와 매개 인자가 없고, url ID만 있다.

<702> 실시 예를 들어 더 자세한 설명을 하겠다.

<703> [실시에 4] 다음은 BitWrapper를 사용해 이미 압축된 3차원 메쉬 정보 비트스트림을 URL을 사용해 전송하는 XMT 예제이다. <Header>

<704> ...

<705> <ObjectDescriptor objectDescriptorID="12" binaryID="12">

<706> <Descr>

<707> <esDescr>

<708> <ES\_Descriptor ES\_ID="PI" binaryID="211" OCR\_ES\_ID="PIC">

<709> <StreamSource>

<710> <BitWrapperEncodingHints>

<711> <BitWrapper3DMCEncodingHints>

<712> <sourceFormat>

<713> <param value="bunny-15000-tcp.m3d"> </param>

<714> </sourceFormat>

<715> </BitWrapper3DMCEncodingHints>

<716> </BitWrapperEncodingHints>

```
<717>    </StreamSource>

<718>    </ES_Descriptor>

<719>        </esDescr>

<720>        </Descr>

<721> </ObjectDescriptor>

<722> </Header>

<723> <Body>

<724>    <Replace>

<725>        <Scene>

<726>    <Group>

<727>        <children>

<728>    <Shape>

<729>        <geometry DEF="MY_BOX">                <BitWrapper type="0" url="12">

<730>    <node>

<731>        <IndexedFaceSet>

<732>            <coord DEF="Box-COORD"></coord>

<733>            <Coordinate> </Coordinate>

<734>        </IndexedFaceSet>                </node>

<735>            </BitWrapper>
```

```

<736>          </geometry>          </Shape>

<737>    </children>

<738>  </Group>

<739> </Scene>

<740> </Replace>

<741> <ObjectDescriptorUpdate>

<742>  <OD>

<743>    <ObjectDescriptorBase>

<744>      <ObjectDescriptor objectDescriptorID="12" binaryID="12">      <URL URLstring="
        UrlWithoutEP.scr">

<745>        </URL>

<746>      </ObjectDescriptor>

<747>    </ObjectDescriptorBase>

<748>  </OD>

<749> </ObjectDescriptorUpdate>

<750> </Body>

<751>    도 2에서 실시예 4와 같이 BitWrapper 노드를 사용해 이미 압축된 3차원 메쉬 정보 비트
스트림을 URL로 전송하는 XMT 표현의 입력 파일이 들어오면, XMTRef 파서(Parser)에서
BitWrapper 노드 안의 URL로 전송되는 압축된 3차원 메쉬 정보 비트스트림을 만들어 낸다. 이

```

것은 BitWrapper 노드와 매개 인자에 대한 XMT-A 스키마 정의를 따른다. 또한 XMT-A 스키마와 스타일 쉬트(XMT2BIFS와 XMT2MUX) 파일을 사용해, MPEG-4 부호화기의 입력 파일(".scene" 파일과 ".mux" 파일)을 만들어 낸다. 이 결과는 다음과 같다.- URLWithoutEP.scene 결과 -

```
<752>      ...
<753>      BitWrapper {
<754>      node IndexedFaceSet {
<755>      coord DEF Box-COORD Coordinate {
<756>          }
<757>      }
<758>      type 0
<759>      url 12
<760>      }
<761>      ...
<762>      UPDATE OD [
<763>      ObjectDescriptor {
<764>          objectDescriptorID 12
<765>          muxScript UrlWithoutEP.scr}
<766>      ]
<767>      - URLWithoutEP.mux 결과 -
```

```

<768>      ...

<769>      ObjectDescriptor {

<770>          objectDescriptorID  12

<771>          esDescr [

<772>              ES_Descriptor {

<773>                  ES_ID    211

<774>          ...

<775>          muxInfo MuxInfo{

<776>              fileName  "bunny-15000-tcp.m3d"

<777>              streamFormat 3DMC

<778>          }

<779>      ...

<780>      위와 같은 scene 파일과 mux 파일은 MPEG-4 부호화기의 입력으로 사용된다. scene 파일
      은 BIFS 부호화기를 거쳐 .bifs/.od 파일을 생성하고, 이 .bifs/.od 파일과 mux 파일이 MP4 부
      호화기에 입력으로 들어가 하나의 비트스트림(".mp4") 파일을 생성한다. 상기 "mp4" 파일은 최
      종 결과로서 MPEG-4 플레이어에서 시각화된다. 본 발명은 컴퓨터로 읽을 수 있는 기록 매체에
      컴퓨터(정보 처리 기능을 갖는 장치를 모두 포함한다)가 읽을 수 있는 코드로서 구현하는 것이
      가능하다. 컴퓨터가 읽을 수 있는 기록 매체는 컴퓨터 시스템에 의하여 읽혀질 수 있는 데이터
      가 저장되는 모든 종류의 기록 장치를 포함한다. 컴퓨터가 읽을 수 있는 기록 장치의 예로는
      ROM, RAM, CD-ROM, 자기 테이프, 플로피 디스크, 광데이터 저장장치 등이 있다.

```

<781> 본 발명은 도면에 도시된 실시예를 참고로 설명되었으나 이는 예시적인 것에 불과하며, 본 기술 분야의 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시예가 가능하다는 점을 이해할 것이다. 따라서, 본 발명의 진정한 기술적 보호 범위는 첨부된 등록청구범위의 기술적 사상에 의해 정해져야 할 것이다.

# 【발명의 효과】

<782> 본 발명에 의한 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성방법 및 시스템에 의하면, 저작자가 메타 표현 방법을 사용해서 3D 콘텐츠 저작단계에서 3D 데이터의 표현 및 압축을 할 수 있다. 저작자가 압축의 선택과 조절이 가능하다. 그 방법은 메타 표현의 선택 여부에 따라 압축의 가능 여부를 결정할 수 있다. 또한 메타 표현 방법을 조절하므로 압축 방식을 조절할 수 있다(encoding paramters, in-band/out-band scenario).

<783> 그리고 본 발명을 이용하여 XML 기반의 MPEG-4 프리미티브(primitive)들을 사용해 쉽게 저작할 수 있는 툴(tool)을 만들 수 있으며, 이렇게 만들어진 2D, 3D 콘텐츠는 어느 플랫폼(platform)의 애플리케이션에서든지 재사용이 가능하다. 또한 저작 단계에서 저작자가 3D 데이터를 압축할 수 있기 때문에 낮은 네트워크 대역폭(bandwidth)에서도 3D 그래픽 데이터의 실시간 시각화 또는 실시간 애니메이션을 가능하게 할 수 있다. 그리고 다양한 3D 그래픽 데이터를 제작할 수 있게 해 준다.

【특허청구범위】

【청구항 1】

적어도 압축할 객체 데이터에 관한 정보를 포함하는 압축노드와, 압축에 필요한 매개인자를 정의하고 있는 XML 스키마를 구비하는 단계;

XML 입력파일을 상기 XML 스키마를 참조하여 소정의 데이터 압축부호화기에 입력되는 파일로의 변환을 지원하는 스타일 쉬트(style sheet)를 구비하는 단계; 및

XML 입력파일을 상기 XML 스키마 및 상기 스타일 쉬트를 참조하여 파싱하여 소정의 데이터 압축부호화기에 입력되는 파일을 생성하는 단계를 포함함을 특징으로 하는 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성방법.

【청구항 2】

제1항에 있어서, 상기 XML 스키마는

적어도 상기 압축된 객체 데이터를 저장하고 있는 파일의 위치정보를 포함하고 있는 EncodingHints를 더 구비함을 특징으로 하는 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성방법.

【청구항 3】

제1항 또는 제2항에 있어서, 상기 압축매개인자는

압축 대상이 되는 객체의 정점 좌표에 대한 키프레임 애니메이션 데이터에 관한 매개인자, 3차원 메쉬 정보에 관한 매개인자, 회전 이동 키프레임 애니메이션 데이터에 관한 매개인자 및 위치 이동 키프레임 애니메이션 데이터에 관한 매개인자 중 적어도 하나를 포함함을 특징으로 하는 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성방법.

## 【청구항 4】

압축할 객체 데이터에 관한 정보를 포함하고 있는 압축노드와, 압축에 필요한 매개인자, 및 적어도 압축된 객체 데이터 파일의 위치정보를 포함하는 BitWrapperEncodingHints를 정의하고 있는 XMT 스키마를 구비하는 스키마단계;

XMT 입력파일을 상기 XMT 스키마를 참조하여 scene 파일로의 변환을 지원하는 XMT2BIFS style sheet와, XMT 입력파일을 상기 XMT 스키마를 참조하여 mux 파일로의 변환을 지원하는 XMT2MUX style sheet를 구비하는 스타일시트단계; 및 XMT 입력파일을 상기 XMT 스키마와 상기 XMT2BIFS style sheet 및 XMT2MUX style sheet를 이용하여 파싱하여 scene 파일 및 mux 파일을 생성하는 파싱단계를 포함함을 특징으로 하는 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성방법.

## 【청구항 5】

제4항에 있어서, 상기 압축노드는 압축할 객체 데이터를 포함하고 있는 노드필드;

url 필드와 상호 배타적으로 사용되며, 상기 노드의 압축된 비트스트림을 인-밴드로서 전송하는 buffer필드; 및

상기 buffer 필드와 상호 배타적으로 사용되며, 상기 노드의 압축된 비트스트림을 아웃-밴드로서 전송하는 url 필드를 포함하여 이루어짐을 특징으로 하는 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성방법.

【청구항 6】

제5항에 있어서, 상기 압축노드는

노드압축 스킴의 종류를 나타내는 타입필드를 더 구비함을 특징으로 하는 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성방법.

【청구항 7】

제4항에 있어서, 상기 압축매개인자는

압축 대상이 되는 객체의 정점 좌표에 대한 키프레임 애니메이션 데이터에 관한 매개인자, 3차원 메쉬 정보에 관한 매개인자, 회전 이동 키프레임 애니메이션 데이터에 관한 매개인자 및 위치 이동 키프레임 애니메이션 데이터에 관한 매개인자 중 적어도 하나를 포함함을 특징으로 하는 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성방법.

【청구항 8】

제4항에 있어서, 상기 BitWrapperEncodingHints는

압축노드의 URL ID 와 같은 객체기술 ID정보 및 mux 파일이 제공하는 압축된 비트스트림을 전송하는 파일이름과 비트스트림 포맷 종류에 관한 정보를 더 구비함을 특징으로 하는 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성방법.

【청구항 9】

제4항에 있어서, 상기 파싱단계는

원본데이터와 압축 매개인자 및 버퍼를 구비하는 압축노드를 포함하는 XMT 입력파일을 받아들이는 단계; 및

상기 XMT 스키마와 상기 XMT2BIFS style sheet 및 XMT2MUX style sheet를 이용하여 상기 XMT 입력파일을 파싱하여 scene 파일 및 mux 파일을 생성하는 단계를 포함하여 이루어짐을 특징으로 하고,

상기 scene 파일은

상기 원본데이터와, 압축 매개인자와, 상기 원본데이터를 압축한 비트스트림을 전송할 버퍼를 구비하며,

상기 mux 파일은

상기 scene 파일을 BIFS 인코더를 통해 부호화된 파일이름과, 스트림 포맷 정보를 구비하는, 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성방법.

#### 【청구항 10】

제4항에 있어서, 상기 파싱단계는

이미 압축된 객체데이터가 일시 저장되어 있는 버퍼정보를 구비하는 압축노드를 포함하는 XMT 입력파일을 받아들이는 단계; 및

상기 XMT 스키마와 상기 XMT2BIFS style sheet 및 XMT2MUX style sheet를 이용하여 상기 XMT 입력파일을 파싱하여 scene 파일 및 mux 파일을 생성하는 단계를 포함하여 이루어짐을 특징으로 하고,

상기 scene 파일은

객체데이터를 압축한 비트스트림을 전송할 버퍼 정보를 구비하며,

상기 mux 파일은

상기 scene 파일을 BIFS 인코더를 통해 부호화된 파일이름과, 스트림 포맷 정보를 구비하는, 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성방법.

【청구항 11】

제4항에 있어서, 상기 파싱단계는

원본데이터와 압축 매개인자 및 url정보를 구비하는 압축노드와, 압축노드의 url ID와 같은 객체기술 ID(objectDescriptor ID) 정보 및 객체의 압축된 비트스트림의 위치정보를 구비하는 BitWrapperEncodingHints를 포함하는 XMT 입력파일을 받아들이는 단계; 및

상기 XMT 스키마와 상기 XMT2BIFS style sheet 및 XMT2MUX style sheet를 이용하여 상기 XMT 입력파일을 파싱하여 scene 파일 및 mux 파일을 생성하는 단계를 포함하여 이루어짐을 특징으로 하고,

상기 scene 파일은

상기 원본데이터와, 압축 매개인자와, 상기 원본데이터를 압축한 비트스트림에 관한 정보를 링크하는 url 정보를 구비하며,

상기 mux 파일은

상기 BitWrapperEncodingHints에 명시된 객체의 압축된 비트스트림의 위치정보 및 스트림 포맷 정보를 구비하는, 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성방법.

【청구항 12】

제11항에 있어서, 상기 XMT 입력파일은

상기 BitWrapperEncodingHints에 나타난 객체 ID와 동일한 객체 기술 ID 정보와 파싱 결과 생성될 mux 파일이름 정보를 포함하고 있는 ObjectDescriptorUpdate를 더 구비하며,

상기 scene 파일은

상기 BitWrapperEncodingHints에 나타난 객체 ID와 동일한 객체기술 ID 정보와 mux 파일 이름 정보를 더 구비함을 특징으로 하는 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성방법.

### 【청구항 13】

제4항에 있어서, 상기 파싱단계는

이미 압축된 객체데이터에 관한 정보와 링크된 URL 정보를 구비하는 압축노드와 URL ID와 같은 객체기술(objectDescriptor) ID 정보 및 객체의 압축된 비트스트림의 위치정보를 구비하는 BitWrapperEncodingHints 를 포함하는 XMT 입력파일을 받아들이는 단계; 및

상기 XMT 스키마와 상기 XMT2BIFS style sheet 및 XMT2MUX style sheet를 이용하여 상기 XMT 입력파일을 파싱하여 scene 파일 및 mux 파일을 생성하는 단계를 포함하여 이루어짐을 특징으로 하고,

상기 scene 파일은

상기 압축노드에 명시된 객체기술 ID와 동일하며 상기 원본데이터를 압축한 비트스트림에 관한 정보와 링크하는 url 정보를 구비하며,

상기 mux 파일은

상기 BitWrapperEncodingHints에 명시된 객체의 압축된 비트스트림의 위치정보 및 스트림 포맷 정보를 구비하는, 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성방법.

【청구항 14】

제13항에 있어서, 상기 XMT 입력파일은

상기 BitWrapperEncodingHints에 나타난 객체 ID와 동일한 객체 기술 ID 정보와 파싱 결과 생성될 mux 파일이름 정보를 포함하고 있는 ObjectDescriptorUpdate를 더 구비하며,

상기 scene 파일은

상기 BitWrapperEncodingHints에 나타난 객체 ID와 동일한 객체 기술 ID 정보와 mux 파일이름 정보를 더 구비함을 특징으로 하는 그래픽 데이터 압축에 관한 메타표현을 이용한 입력 파일 생성방법.

【청구항 15】

제1항 내지 제14항 중 어느 한 항에 기재된 발명을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체.

【청구항 16】

적어도 압축할 객체 데이터에 관한 정보를 포함하는 압축노드와, 압축에 필요한 매개인자를 정의하고 있는 XML 스키마;

XML 입력파일을 상기 XML 스키마를 참조하여 소정의 데이터 압축부호화기에 입력되는 파일로의 변환을 지원하는 스타일 쉬트(style sheet); 및

XML 입력파일을 상기 XMT 스키마 및 상기 스타일 쉬트를 참조하여 파싱하여 소정의 데이터 압축부호화기에 입력되는 파일을 생성하는 XLM파서를 포함함을 특징으로 하는 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성시스템.

**【청구항 17】**

제16항에 있어서, 상기 압축매개인자는

압축 대상이 되는 객체의 정점 좌표에 대한 키프레임 애니메이션 데이터에 관한 매개인자, 3차원 메쉬 정보에 관한 매개인자, 회전 이동 키프레임 애니메이션 데이터에 관한 매개인자 및 위치 이동 키프레임 애니메이션 데이터에 관한 매개인자 중 적어도 하나를 포함함을 특징으로 하는 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성시스템.

**【청구항 18】**

압축할 객체 데이터에 관한 정보를 포함하고 있는 압축노드와, 압축에 필요한 매개인자, 및 적어도 압축된 객체 데이터 파일의 위치정보를 포함하는 BitWrapperEncodingHints를 정의하고 있는 XMT 스키마;XMT 입력파일을 상기 XMT 스키마를 참조하여 scene 파일로의 변환을 지원하는 XMT2BIFS 스타일 쉬트;

XMT 입력파일을 상기 XMT 스키마를 참조하여 mux 파일로의 변환을 지원하는 XMT2MUX 스타일 쉬트; 및

XMT 입력파일을 상기 XMT 스키마와 상기 XMT2BIFS style sheet 및 XMT2MUX 스타일 쉬트를 이용하여 파싱하여 scene 파일 및 mux 파일을 생성하는 XMT파서를 포함함을 특징으로 하는 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성시스템.

**【청구항 19】**

제18항에 있어서, 상기 압축노드는

압축할 객체 데이터를 포함하고 있는 노드필드;

url 필드와 상호 배타적으로 사용되며, 상기 노드의 압축된 비트스트림을 인-밴드로서 전송하는 buffer 필드; 및

상기 buffer 필드와 상호 배타적으로 사용되며, 상기 노드의 압축된 비트스트림을 아웃-밴드로서 전송하는 url 필드를 포함하여 이루어짐을 특징으로 하는 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성 시스템.

【청구항 20】

제18항에 있어서, 상기 압축매개인자는

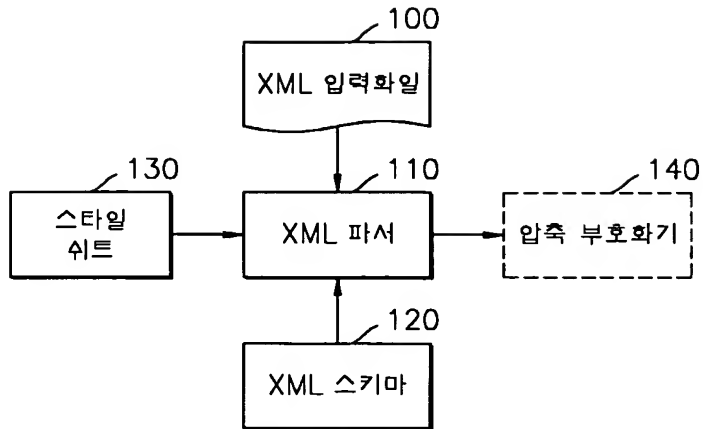
압축 대상이 되는 객체의 정점 좌표에 대한 키프레임 애니메이션 데이터에 관한 매개인자, 3차원 메쉬 정보에 관한 매개인자, 회전 이동 키프레임 애니메이션 데이터에 관한 매개인자 및 위치 이동 키프레임 애니메이션 데이터에 관한 매개인자 중 적어도 하나를 포함함을 특징으로 하는 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성시스템.

【청구항 21】

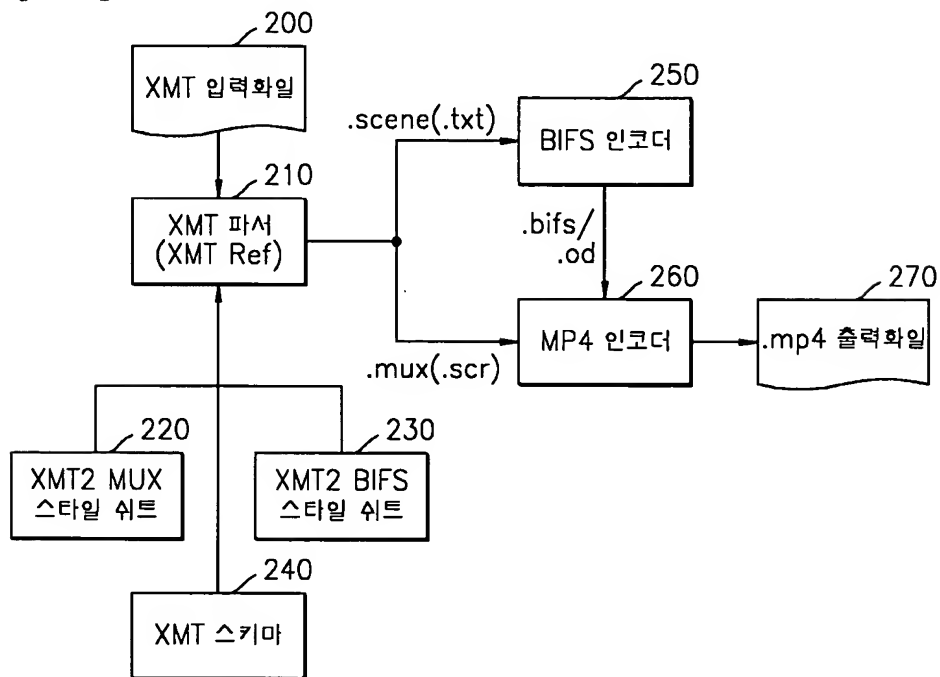
제18항에 있어서, 상기 BitWrapperEncodingHints는 압축노드의 URL ID와 같은 객체기술(objectDescriptor) ID 정보 및 mux 파일이 제공하는 압축된 비트스트림을 전송하는 파일이름과 비트스트림 포맷 종류에 관한 정보를 더 구비함을 특징으로 하는 그래픽 데이터 압축에 관한 메타표현을 이용한 입력파일 생성시스템.

## 【도면】

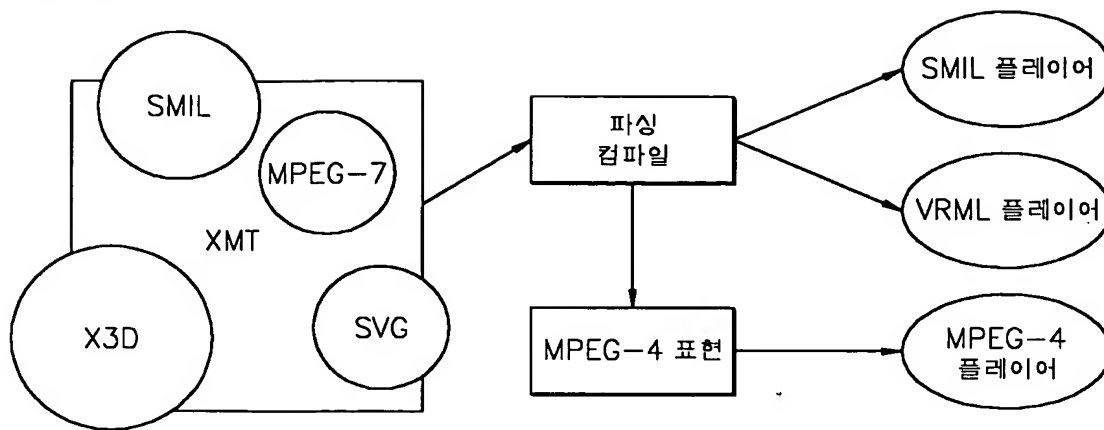
【도 1】



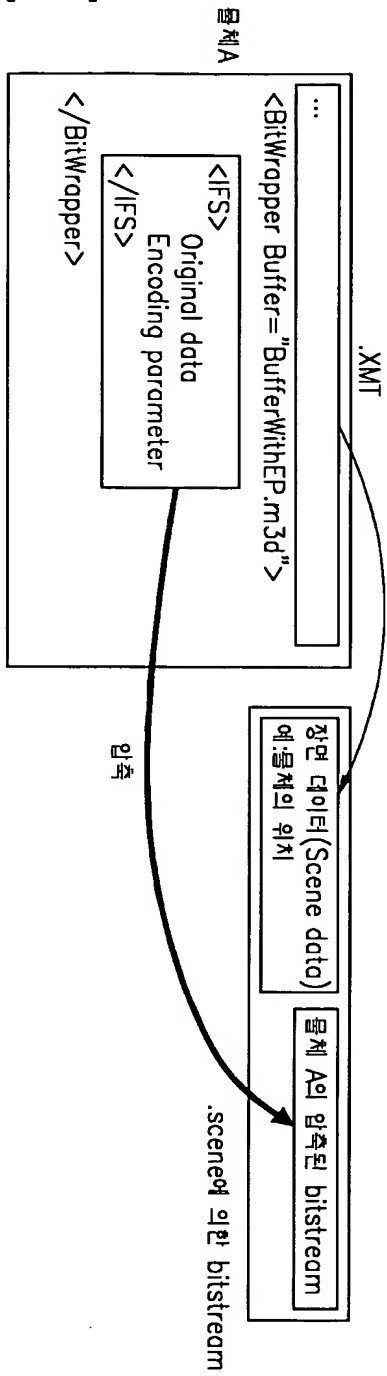
【도 2】



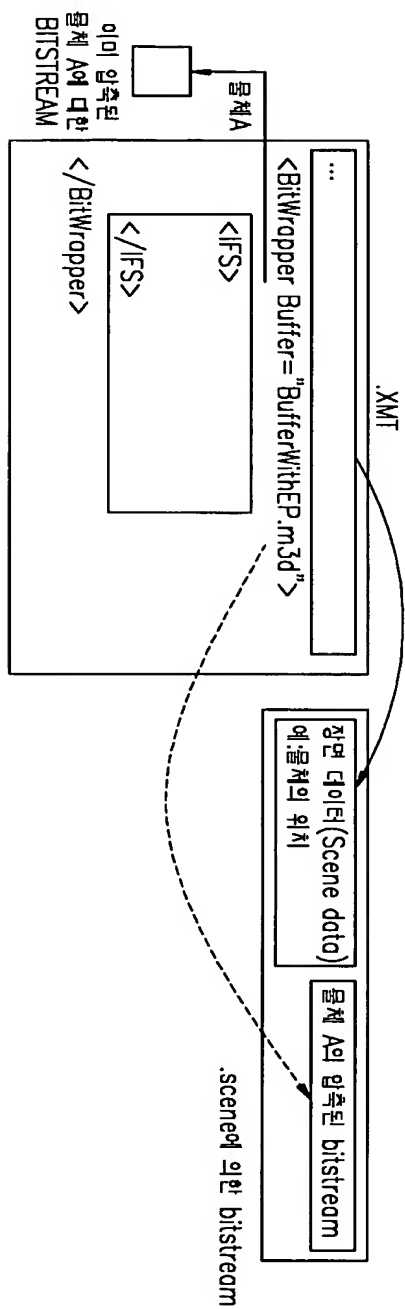
【도 3】



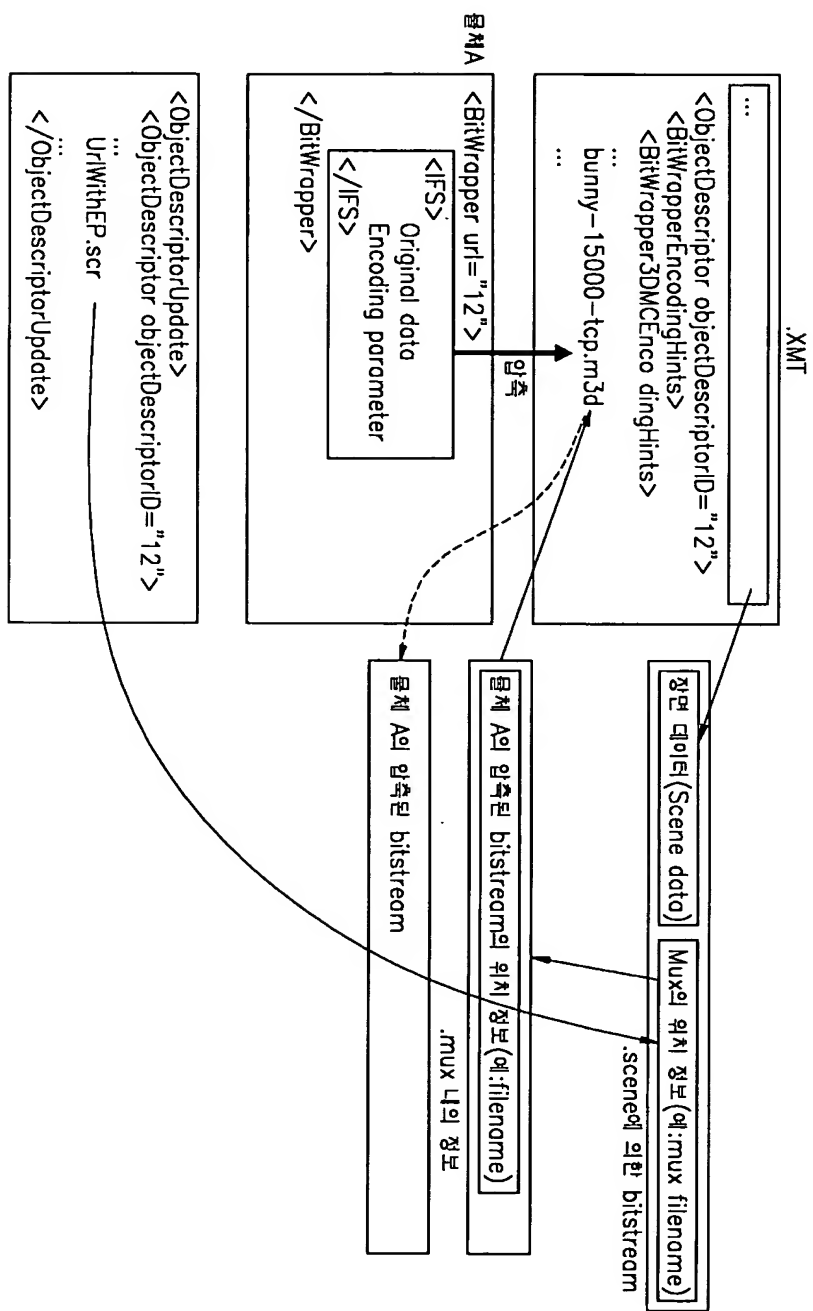
【도 4】



【표 5】



【도 6】



【도 7】

